Oak Street
UNCLASSIFIED

GCA

STANDARD 101-1983

Document Markup Metalanguage

Adopted by
DEPARTMENT OF DEFENSE
10 August 1983

GENCODE™

And The

STANDARD GENERALIZED MARKUP LANGUAGE

(S G M L)

GCA

Graphic Communications Association
1730 North Lynn Street / Suite 604 • Arlington, Virginia 22209 • Phone (703) 841-8160

An Affiliate of Printing Industries of America, Inc. PIA INC

GCA

STANDARD 101-1983

Document Markup Metalanguage

Adopted by
DEPARTMENT OF DEFENSE
10 August 1983

GENCODE™

And The

STANDARD GENERALIZED MARKUP LANGUAGE

(S G M L)

GCA

Graphic Communications Association
1730 North Lynn Street / Suite 604 • Arlington, Virginia 22209 • Phone (703) 841-8160

An Affiliate of Printing Industries of America, Inc.

GCA Std 101-1983
10 August 1983

ACCEPTANCE NOTICE

This nongovernment document was adopted on 10 August 1983 and is approved for use by DoD. The Graphic Communications Association has furnished the clearance required by existing regulations. Copies of this document are stocked by DoD Single Stock Point, Naval Publications and Forms Center, Philadelphia, PA 19120, for issue to DoD activities only. Contractors and industry groups must obtain copies directly from the Graphic Communications Association, 1730 North Lynn Street, Suite 604, Arlington, VA 22209.

Title of Document:	Information Processing Systems— Programming Languages— Text Interchange and Processing
Date of Specific Issue Adopted:	1 June 1983
Releasing Industry Group:	Graphic Communications Association (An Affiliate of Printing Industries of America, Inc.)

NOTICE: When reaffirmation, amendment, revision, or cancellation of this standard is initially proposed, the Graphic Communications Association shall inform the Military Coordinating Activity of the proposed change and request participation. The primary purpose of this document is the promulgation, for trial use, of the standard as submitted to the International Organization for Standardization (ISO) for coordination internationally. Certain portions are still in development and are not approved for use by the Department of Defense. Accordingly, the document should not be referenced in toto, and specific references should be tailored to specific acquisitions. **Only Parts One and Six (see references on following page) are authorized for use by DoD.**

NOTICE: This standard is a TRIAL-USE STANDARD subject to the limitations expressed on the following page.

Custodians:

Army— MI
Navy— AS
Air Force—02

Military Coordinating Activity:

Air Force—02
(Project IPSC—0205)

User activities:

Army— AD, TA, MI
Navy— OM, SH
Air Force—02, 16

AREA IPSC

University of
Illinois Library
at Urbana-Champaign
Oak Street

GCA Std 101-1983

Information Processing Systems—
Programming Languages—
Text Interchange and Processing

Preliminary Trial-Use Standard

PURPOSE

The purpose of this document is to provide the domestic and international printing and publishing industries with access to this standard, for trial-use during the process of review and adoption of it by the International Organization for Standardization (ISO).

REFERENCES

ISO draft standard documents and working draft documents comprising the trial-use standard include:

TC97/SC5/EG CLPT--X3J6 Part One: General	N177-1 Sixth Working Draft	*See Note*
TC97/SC5/EG CLPT--X3J6 Part Six: Document Markup Metalanguage	N177-6 Sixth Working Draft	*See Note*

NOTE: Proposed as ISO Draft Proposed International Standard.

DEVELOPMENT

This standard was developed under the regular policies, organization, and procedures of:

1. **American National Standards Committee X3J6**

   X3 — Information Processing Systems
   J6 — Computer Languages for the Processing of Text

   Project 203

and 2. **International Organization for Standardization (ISO)**

   TC97/SC5/EG CLPT
   TC97 — Information Processing
   SC5 — Programming Languages
   EG — Experts Group
   CLPT — Computer Languages for the Processing of Text

   Project 97.5.12

NOTE

The provisions included within this document may change as a result of the review and coordination process within the International Organization for Standardization (ISO) and the American National Standards Committee X3, Information Processing Systems.

LIMITATION

Upon formal publication of the accepted standard by the International Organization for Standardization and the American National Standards Institute, the Graphic Communications Association will withdraw this preliminary, trial-use standard, and will support the accepted standard.

ISBN: 0-89740-244-8

Library of Congress Catalog Card No. 81858

The Graphic Communications Association
Arlington, Virginia 22209

© 1983 by the Graphic Communications Association.
All rights reserved.
Published 1983.
First impression 1983.
Printed in the United States

Special Note

This copyright does NOT extend to any page of this book identified as a page of the "Sixth Working Draft, International Standard ISO TC97/SC5/EG CLPT - X3J6 N177 1983 June 01."

Foreword

It is with a great deal of satisfaction that the Graphic Communications Association greets the publication of Standard Generalized Text Markup Language. The publication culminates nearly 16 years of effort on the part of the organization to develop the concepts that it entails and to encourage their acceptance in the industry.

William W. Tunnicliffe introduced those concepts to the organization in 1967. As member representative from the Courier Corporation, one of the Founders of GCA, he served as a Director and as Chairman of our Character Generation Committee. At that time, computerized composition, while growing fast, was in many respects very much in its infancy. For input, facilities relied upon either old electromechanical technologies or electronic devices designed for general data processing. ''Data bases,'' as often as not, consisted of stretches of paper tape. Even those that were in magnetic form were limited to the typesetting image for which they were coded. Type was the intended product of the system, and profits were elusive at best.

Text markup followed the lead of hot metal: it described output formats. The competitive drive was to develop the system that facilitated the markup and keyboarding process within this output orientation. Any talk of ''standards'' then, and for too many years thereafter, focused on standardizing typesetting commands. Needless to say, the industry response to generic coding for markup, with a focus on the textual data base and the associated editorial structure, was severely limited. We were encouraged, however, by the Internal Revenue Service which, led by Electronic Publications Specialist William W. Davis, Jr., saw the many values of the concepts early on; by then Deputy, and later Public Printer, John J. Boyle, now retired who was a working participant in our committee sessions; by the work of Dr. Charles F. Goldfarb of IBM who learned of our concepts at our 1971 Spring Conference; by the achievements of the American Chemical Society whose Chemical Abstracts Service, under the leadership of Dr. Fred A. Tate, Associate Director, and Dr. Ronald L. Wigington, Director of Research and Development, established what was probably the first true system implementation for generic coding; by the perennial striving of X3/SPARC, led by Charles D. Card, Staff Director, System Standards, Sperry Computer Systems, to define the need for standards in the text arena; and, in the mid-70's, by the moral support of Ms. Sarah N. Rhodes, then Assistant Program Director for the Information Science Program of the National Science Foundation.

Through committee work sessions, seminar programs, workshops, and other means, GCA directed its limited resources to the development of generic coding. From all of this emerged, in the late 70's, GenCode™, with its concept of a document, its inclusion of the delineation of text hierarchies along with generic coding, its emphasis on content-driven product as opposed to form-driven product, and other key factors that facilitated its application to what, hopefully, will be the full range of text structures.

The GenCode™ concept is applications oriented. Yet, to be fully effective and bring to the using community the widest range of benefits possible, the application must link to a recognized uniform structure for computer processing. The writer, a charter member of the SPARC/Text Committee, and other GCA representatives have long taken part in the ANSI X3 text effort, including participation in the organizational meeting from which came the statement of purpose and direction for X3J6. It was our hope that the underlying system could emerge as a national and international standard. Our GenCode™ Committee provided its full support to the X3 effort, and Aaron H. Bigman, then GCA Vice President, Information Technologies, presented our concepts at the meeting that brought about ISO recognition of the GenCode™ approach to text documentation.

GCA's membership encompasses a broad array of publishers—commercial, industrial, professional societies, governmental—with products ranging from trade books, magazines, and newspapers, to reference manuals, catalogs, technical documentation, law books, and so forth; computer composition and data processing service facilities; printers; and software and equipment suppliers. We hope we will begin the move toward adoption of GenCode™ and the Standard Generalized Markup Language. We particularly encourage their comments on this publication and their recommendations on the standard including its application to tabular, mathematical, and graphic representations.

[BLANK PAGE]

Preface

This standard is a tribute to a dedicated group of individuals whose enduring cohesiveness is matched by their individual abilities, persistence, stamina, uninhibited candor during technical discussions, and lasting good humor. It is, to be sure, a record of their years-long work, but, more importantly, it is a *practical working standard that will allow—and encourage—practical system implementation*. It took significant intellectual courage to approach the development of the standard dispassionately, reviewing the technical environment in proper perspective and resisting the sirens of immediate cookbook implementation that have characterized some earlier efforts. The deliberate, considered choice of establishing a syntactical legitimacy that would provide a solid, logical foundation for a lasting, logical implementation has resulted in a strong, but flexible, standard. Because it reflects a unique admixture of existing operational practice, theory, and thought, the standard will stand the test of time to become a firm base for future refinements.

This product is the result of the cooperative spirit of an industry group, the GenCode™ Committee of the Graphic Communications Association, that became, simultaneously, the body of the Text Description Task Group, which was concurrently ANSI X3J6.1 and ISO TC97/SC5 EG CLPT/TG. The Task Group made the conscious, and courageous, choice to work on the development of the standard by the consensus method within the duly constituted standards development process defined and maintained by the two cognizant standards organizations: the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO).

Every member of the standard development group has made noteworthy contributions to both the substance of the standard and the group's enduring spirit. Each member joins me in giving special recognition to Charles D. Card, Chairperson of ISO TC97/SC5 EG CLPT and of ANSI X3J6, who provided the proper environment for the markup language and sagaciously steered the Text Description Task Group through what were, for them, the uncharted waters of the standards development process; to William W. Davis, Jr., Chairperson of the Text Description Task Group, whose creative and effective leadership was vital to completion of the standard, and to Charles F. Goldfarb, Editor-in-Chief, whose creative and comprehensive contributions encouraged continuity, correlation, control, and technical accuracy in the generation and production of the many working drafts preceding the official submission copy.

A very special measure of recognition must be given to Norman W. Scharpf, President of the Graphic Communications Association. His foresight and vision in assessing the potential usefulness of generic coding; his encouragement of early efforts to define, and continuing efforts to disseminate, the root concepts through sponsored Work Shops, Annual Conferences, and GenCode™ Conferences; his resolute steadfastness in supporting GenCode™ Committee efforts through the long, lean years; his constant provision of a forum for, and his own individual contributions to, technical progress; and his support of GenCode™ Committee Member participation in the formal ANSI/ISO standard development process have given the effort its substance, the standard its special meaning.

Book Organization

The *Document Markup Metalanguage Standard* also referred to as the *Standard Generalized Markup Language* (*SGML*) is presented on pages 1 through 98 as Part A in this book. Pages 1 through 80 present the provisions submitted for the official standard while Annexes A through D, which appear on pages 81 through 98, provide supporting material. These 98 pages constitute Part Six of the entire standard while Part One, which appears in the specially-added Annex Z, pages 99 through 111, places Part Six in contextual reference to the entire standard. Only Parts One and Six are included in this publication; the Computer and Business Equipment Manufacturers Association (CBEMA) is the X3 Secretariat and can provide copies of Parts Two through Five (see page 103 for additional information). Parts Six through Ten are still under development.

The material presented on pages 1 through 112 constitutes GCA Standard 101-1983 and, concurrently, is the exact text created through the standards development process. This text, together with the Sixth Working Draft of Parts Two through Five, was officially submitted to ISO on 15 June 1983 through ANSI, which is the Secretariat for TC97/SC5, by Charles D. Card, Chairperson of EG CLPT, the Experts Group for Computer Languages for the Processing of Text.

Most especially we enlist the support of the total publishing community toward the next step in GenCode™ development, the creation and the registration of standard document type definitions.

Consistent with its history, GCA stands ready to assume a leadership role, in concert with governmental, commercial, and industry groups who share our interest in this area, in defining and implementing the courses of action which will result in widespread use of Standard Generalized Markup Language system implementations.

NORMAN W. SCHARPF
President
Graphic Communications Association

SUMMARY TABLE OF CONTENTS

PART A

GCA Standard 101-1983
Document Markup Metalanguage (SGML)

0. Introduction 1
1. Scope and Field of Application 5
2. Basic Concepts 6
3. Extended Concepts 34
4. Formal Specification 52
5. Conformance to the Standard 78

Annexes

A. Introduction to Generalized Markup 81
B. Implementation Considerations 89
C. Markup Construct Registration 94
D. Nonconforming Variations 97
Z. Part One: General 99

PART B

GenCode™ and the Standard Generalized Markup Language (SGML)

1. **GenCode™ Model Contract Specification** 117

a. **"Internal Revenue Service Specifications for Employment of Generic Coding—Three Levels of Implementation,"** 1982 Nov 24 119

William W. Davis, Jr.
Computer Specialist/Procurement
Publishing Services Branch
Internal Revenue Service, and
Chair, X3J6.1

(1) Scope 127
(2) Term of Contract 127
(3) Basis of Award 127
(4) Contract Administration Contact 128
(5) Order Procedure 128
(6) Schedule 129
(7) Pickup and Delivery of Material 129
(8) Security 129
(9) Supplies 130
(10) Proofreading 130
(11) Description of General Requirements and IRS Standards 130
(12) Description of Composition Requirements 133
(13) Description of Composition Related Requirements ... 135
(14) Explanation of Charges 136
(15) Schedule of Charges 138
(16) Quality Factors Capabilities 141
(17) Definitions 141

In case of conflict between any statement in any article and the standard, the standard governs.

Part B of the book, which starts on page 113, contains three sections of expository material serving as an aid to understanding the generic coding concepts that form the basis of the Document Markup Metalanguage Standard (Standard Generalized Markup Language (SGML)). The first section, entitled ''Model Contract Specification,'' was developed by William W. Davis, Jr., for use by the Internal Revenue Service. The second section is the ''GenCode™ Primer,'' which was originally developed by the GenCode™ Committee prior to its intensive involvement in the creation of the standard. The third section contains articles prepared by individual authors: ''An Introductory Rationale for the Standard Generalized Markup Language,'' which was written specifically for this book, and ''A Short Introduction to Office Applications of Markup Languages—Using SGML,'' which was prepared for the ISO TC97/SC18/WG3 international meeting in Paris 12-15 April 1983, both bear directly on the standard. The remainder deal with the concept of, and experiences with, generic coding. These essays describe practices of different users that are not always in direct compliance with the provisions of the standard. They do, however, provide helpful insights into the practical dimensions of system implementations made before the standard was created.

I would also like to assure the reader that, despite the ostensible complexity of the standard, generic codes, GenCode™, and SGML in system implementation form (i.e., with the preprocessor written) are most easily used. One approach to absorbing the material presented is: (1) to scan through the standard in Part A to gain an understanding of its structure and scope, (2) to read the expository and example material in Part B to gain a sense of practical operations and the impact that these concepts are having on text handling and computerized composition in everyday, real-world applicataions, and (3) to reread the standard in detail to appreciate the syntactical requirements being placed upon the preprocessor for the different levels of implementation.

William W. Tunnicliffe
Vice-President
Information Technologies
Graphic Communications Association

Washington
August 1983

PART B (*Continued*)

GenCodeⓉⓂ and the Standard Generalized Markup Language (SGML)

2. **GenCodeⓉⓂ Primer** 165

   I. Introduction 171
   II. How GenCodeⓉⓂ Works 172
   III. Implementation Considerations 177

3. **GenCodeⓉⓂ Expository Material** 183

   a. **"An Introductory Rationale for the Standard Generalized Markup Language,"** 1983 June 15 183

   C. "Kit" von Sück
   Marketing Specialist
   Publishing & Broadcast Industries
   Digital Equipment Corporation

   b. **"A Short Introduction to Office Applications of Markup Languages—Using SGML,"** 1983 April 15 187

   James D. Mason
   Text Management Specialist
   Oak Ridge National Laboratory

   c. **"Generic Codes,"** 1978 193

   A collective effort under the leadership of:

   Brian L. Murray, Director
   Text Editing & Photocomposition Branch
   John A. Sullivan, Head
   Composition Software Section
   Canadian Government Printing Office

   d. **"Full-Text Data Base Publishing,"** 1983 May 12 201

   Donald H. Rollert
   Manager, Graphic Systems Development Division
   U.S. Government Printing Office

   e. **"Integrating Text and Graphics,"** 1982 213

   Aaron H. Bigman
   Senior Financial Publishing Consultant
   Standard & Poor's Corporation

   f. **"Rules & Regulations Via GenCodeⓉⓂ,"** 1983 May 12 221

   Joseph P. Johnson
   Technology Assessment Officer
   Federal Deposit Insurance Corporation

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

PART B (*Continued*)

GenCode™ and the Standard Generalized Markup Language (SGML)

GenCode™ Expository Material (*Continued*)

g. **"GenCode™ Preprocessor Requirements,"** 1983 June 15 227

Jordan J. Glogau
Executive Vice President
Medianet, Inc.

h. **"Document Processing at Boeing Computer Services,"** 1983 June 15 243

Sharon Adler
Manager, Production Services
Boeing Computer Services

i. **"Comparison of Generic, Mnemonic, and Native-Mode Coding,"** 1983 May 3 247

Joseph V. Gangemi
Vice President, Systems and Programming
Lehigh/Rocappi, Inc.

SUMMARY TABLE OF CONTENTS

PART C

Contributors

1. ANSI/ISO Committee Participation 255
2. Authors' Biographies 260

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

GCA

STANDARD 101-1983

Document Markup Metalanguage

(S G M L)

Adopted by
DEPARTMENT OF DEFENSE
10 August 1983

GCA

Graphic Communications Association
1730 North Lynn Street / Suite 604 • Arlington, Virginia 22209 • Phone (703) 841-8160

An Affiliate of Printing Industries of America, Inc.

[BLANK PAGE]

Part Six: Document Markup Metalanguage

Dr. Charles F. Goldfarb
Editor-in-Chief

Senior Technical Advisor
IBM Corporation

Member		
ANSI		X3J6
Editor		
ANSI	Task Group	X3J6.1
EG CLPT	Task Group	SGML

[BLANK PAGE]

Information Processing Systems - Programming Languages - Text Interchange and Processing - Part Six: Document Markup Metalanguage

Sixth Working Draft, International Standard

ISO TC97/SC5/EG CLPT - X3J6 N177-6 1983 Jun 01

[BLANK PAGE]

Table of Contents

0 Introduction 1
  0.1 Background of the Standard 1
  0.2 Objectives of the Standard 2

1 Scope and Field of Application 5
  1.1 Scope 5
  1.2 Field of Application 5

2 Basic Concepts 6
  2.1 Documents, Type Definitions, and Procedures 6
    2.1.1 Documents 6
    2.1.2 Type Definitions 6
    2.1.3 Procedures 7
  2.2 Markup 7
  2.3 Distinguishing Markup from Text 9
    2.3.1 Descriptive Markup Tags 9
    2.3.2 Other Markup 10
    2.3.3 Record Boundaries 10
      2.3.3.1 Record Boundaries in Text 11
      2.3.3.2 Other Record Boundaries 11
  2.4 Document Structure 11
    2.4.1 Type Definitions 12
    2.4.2 Structure Declarations 12
      2.4.2.1 Connectors and Occurrence Indicators 12
      2.4.2.2 Groups 13
      2.4.2.3 Using Entity References 14
      2.4.2.4 Text Characters 14
      2.4.2.5 Data Content 15
      2.4.2.6 Summary of Model Delimiters 16
  2.5 Attributes 16
    2.5.1 Specifying Attributes 16
      2.5.1.1 Names 17
      2.5.1.2 Attribute Values 17
    2.5.2 Declaring Attributes 18
      2.5.2.1 Attribute Declaration Syntax 18
      2.5.2.2 Complex Attribute Values 19
      2.5.2.3 Changing Default Values 20
  2.6 Entities 20
    2.6.1 Entity Reference Syntax 20
    2.6.2 Declaring Entities 21
      2.6.2.1 Literals 21
      2.6.2.2 Processing Instructions 22
      2.6.2.3 Entities with Entity References 23
      2.6.2.4 External Entities 23
    2.6.3 Entity Sets 23
    2.6.4 Entities in Markup Declarations 23
  2.7 Marked Sections 24
    2.7.1 Ignoring a Marked Section 24
    2.7.2 Unparsable Sections 25
    2.7.3 Temporary Sections 25
    2.7.4 Storing a Marked Section as an Entity 25
    2.7.5 Using Delimiter Characters as Data 26
  2.8 Character Sets 26

[BLANK PAGE]

- 4.4.2 Attribute Specification 57
  - 4.4.2.1 Attribute Value 58
  - 4.4.2.2 Name 59
  - 4.4.2.3 Rank Number 59
  - 4.4.2.4 Separator 59
- 4.5 Entity Reference 59
- 4.6 Parameter Entity Reference 59
- 4.7 Self-delimiting Entity Reference 60
- 4.8 Processing Instruction 60
- 4.9 Markup Declaration 60
  - 4.9.1 Parameter 60
  - 4.9.2 Entity Declaration 61
    - 4.9.2.1 Entity Name 61
    - 4.9.2.2 Entity 62
    - 4.9.2.3 Literal 63
  - 4.9.3 Structure Declaration 63
    - 4.9.3.1 Coexistent Structure 64
    - 4.9.3.2 Element Type 64
    - 4.9.3.3 Markup Minimization 64
    - 4.9.3.4 Declared Content 65
    - 4.9.3.5 Content Model 65
    - 4.9.3.6 Content Exceptions 66
  - 4.9.4 Attribute Declaration 67
    - 4.9.4.1 Name Group 67
    - 4.9.4.2 Markup Minimization 67
    - 4.9.4.3 Declared Value 67
    - 4.9.4.4 Value Model 68
    - 4.9.4.5 Value Exceptions 68
    - 4.9.4.6 Default Value 69
  - 4.9.5 Delimiter Declaration 69
    - 4.9.5.1 Function 69
    - 4.9.5.2 Roles and Assigned Strings 70
- 4.10 Marked Section 73
  - 4.10.1 Keyword Parameters 73
  - 4.10.2 Markup Environments 73
- 4.11 SGML Specification 74
  - 4.11.1 Feature Use 74
  - 4.11.2 Variations 75
    - 4.11.2.1 Quantitative Characteristics 75
    - 4.11.2.2 Character Use 75
    - 4.11.2.3 Delimiter Use 76
    - 4.11.2.4 Data Content Characteristics 76
    - 4.11.2.5 Alias Use 76

5 Conformance to the Standard 78
- 5.1 Conforming Documents 78
  - 5.1.1 Basic SGML Document 78
  - 5.1.2 Extended SGML Document 79
  - 5.1.3 Variant SGML Document 79
- 5.2 Conforming Systems 79
  - 5.2.1 Basic SGML Processor 79
  - 5.2.2 Extended SGML Processor 80
  - 5.2.3 Customizable SGML Processor 80
  - 5.2.4 Variant SGML Processor 80

- 2.9 More About Structure 27
  - 2.9.1 Exceptions to the Model 27
  - 2.9.2 Null Content 28
  - 2.9.3 Alternative Models 28
- 2.10 Data Content 29
  - 2.10.1 Data Content Representations 29
    - 2.10.1.1 Character Set Data (CSDATA) 29
    - 2.10.1.2 Undefined Character Set Data (UCSDATA) . . . 30
    - 2.10.1.3 Bit Data (BITDATA) 30
  - 2.10.2 Data Content Notations 30
    - 2.10.2.1 Character Notations 31
    - 2.10.2.2 Bit Data Notations 31
- 2.11 Customizing 32
  - 2.11.1 The SGML Specification 32
  - 2.11.2 System-issued Declarations 33
- 2.12 Conformance 33

3 Extended Concepts 34
- 3.1 Markup Environments 34
- 3.2 ENTITY: Dynamic Entity Definition 34
- 3.3 STRUC: Dynamic Structure and Attribute Definition 35
- 3.4 DELIM: Delimiter Modification 35
  - 3.4.1 Changing Markup Delimiters 35
  - 3.4.2 Concurrent Delimiter Assignments 35
  - 3.4.3 Selection Criteria 36
  - 3.4.4 Delimiter Environments 36
- 3.5 RECORD: Record Boundary Recognition 37
- 3.6 SDENTITY: Self-delimiting Entity Reference 37
- 3.7 Structural Markup Minimization 38
  - 3.7.1 The Concept of Minimization 38
  - 3.7.2 Specifying Minimization 39
  - 3.7.3 OMITTAG: Omitted Tag Minimization 39
  - 3.7.4 CENDTAG: Contained End-tag Minimization 40
  - 3.7.5 NULLTAG: Null Tag Minimization 41
  - 3.7.6 DATATAG: Data Tag Minimization 42
    - 3.7.6.1 Literals in Models 42
    - 3.7.6.2 Data Tags 43
  - 3.7.7 Structural Markup Minimization Considerations 46
- 3.8 RANK: Explicit Rank 46
- 3.9 ATTMIN: Attribute Minimization 48
- 3.10 ATTEXT: Attribute Extension 49
  - 3.10.1 Complex Value Models 49
  - 3.10.2 Redundant AVC Delimiters 49
- 3.11 ID: Unique Identifier Attribute 50
- 3.12 COSTRUC: Coexistent Structure 51

4 Formal Specification 52
- 4.1 Notation and Conventions 52
- 4.2 Document 52
- 4.3 Element 53
  - 4.3.1 Start-tag 53
  - 4.3.2 Content 54
    - 4.3.2.1 Character 55
  - 4.3.3 End-tag 56
- 4.4 Descriptive Markup 56
  - 4.4.1 Generic Identifier (GI) 57

0 Introduction

This document is Part Six of a multipart standard for Text Interchange and Processing. It defines a metalanguage for document preparation referred to as the "Standard Generalized Markup Language." Unqualified references to "the Standard" refer to the Standard Generalized Markup Language, not to the full multipart standard.

0.1 Background of the Standard

A document can be viewed in the abstract as a structure of various types of element. An author organizes a book into chapters that contain paragraphs, for example, and figures that contain figure captions. An editor organizes a magazine into articles that contain paragraphs that contain words, and so on.

Applications treat these elements in different ways. A formatting application might print headings in a prominent type face, leave space between paragraphs, and otherwise visually convey the structure and other attributes to the reader. An information retrieval system would perhaps assign extra significance to words in a heading when creating its dictionary.

Although this connection between a document's attributes and its processing seems obvious, it tended to be obscured by early text processing methods.

In the days before automated typesetting, an editor would "mark up" a manuscript with the specific processing instructions that would create the desired format when executed by a compositor. Any connection between the instructions and the document's structure was purely in the editor's head.

Early computerized systems continued this approach by adding the process-specific "markup" to the machine-readable document file. The markup still consisted of specific processing instructions, but now they were in the language of a formatting program, rather than a human compositor. The file could not easily be used for a different application, or on a different computer system, without changing all the markup.

As users became more sophisticated, and as text processors became more powerful, approaches were developed that alleviated this problem. "Macro calls" (or "format calls") were used to identify points in the document where processing was to occur. The actual processing instructions were kept outside of the document, in "procedures" (or "macro definitions" or "stored formats"), where they could more easily be changed for different applications.

While the macro calls could be placed anywhere in a document, users began to recognize that most were placed at the start or end of document elements. It was, therefore, natural to choose names for such macros that were "generic identifiers" of the element types, rather than names that suggested particular processing (e.g., "heading" rather than "format-17"), and so the practice of "generic coding" began.

Generic coding was a major step toward making automated text processing systems reflect the natural relationship between document attributes and processing. The advent of "generalized markup languages" in the early 1970's carried this trend further by providing a formal language base for generic coding. A generalized markup language observes two main principles:

1. Descriptive markup predominates and is distinguished from processing instructions.

   Descriptive markup includes both generic identifiers and other attributes of document elements that motivate processing instructions. The processing instructions, which can be in any language, are normally collected outside of the document in procedures.

Annex A. Introduction to Generalized Markup 81
  A.1 The Markup Process 81
  A.2 Descriptive Markup 82
  A.3 Rigorous Markup 85
  A.4 Conclusion 87
  A.5 Acknowledgments 88
  A.6 References 88

Annex B. Implementation Considerations (Not Part of the Standard) 89
  B.1 A Model of the Markup Interpretation Process 89
    B.1.1 Physical Input 89
    B.1.2 Scanning Modes 89
    B.1.3 Markup Minimization 90
    B.1.4 Translation 91
  B.2 Markup Support Functions 91
    B.2.1 Initialization 91
    B.2.2 Structure Processing 92
    B.2.3 Associating GIs and Attributes with Procedures 92
    B.2.4 Functions Available to Procedures 92

Annex C. Markup Construct Registration (Not Part of the Standard) 94
  C.1 Document Type Definitions 94
    C.1.1 Common Document Types 94
    C.1.2 Common Element Types 94
    C.1.3 Pro Forma Document and Element Types 95
  C.2 Data Content Notations 95
  C.3 Entity Sets 95
  C.4 Word Processing Conventions 96

Annex D. Nonconforming Variations (Not Part of the Standard) . 97
  D.1 Fixed-length Generic Identifiers 97
  D.2 Single Delimiter 97

Annex Z. Part One: General 99

Note: Included to provide contextual reference to Parts One through Five of the Standard as a whole.

1. Documents "marked up" with the language must be processable by a wide range of text processing and word processing systems.

   The full form of the language, with all optional features, offers generality and flexibility that can be exploited by sophisticated systems, such as the Text Processing Programming Language specified in Part Three of the Text Interchange and Processing multipart standard. Less powerful systems need not support the features. To facilitate interchange between dissimilar systems, an "SGML specification" describes any markup features or variations used in a document.

2. There must be no character set dependency, as documents might be keyed on a variety of devices.

   The language has no dependency on a particular character set. Any character set that has bit combinations for letters, numerals, space, and delimiters is acceptable.

3. There must be no application, system, or device dependencies.

   Generalized markup is predominantly descriptive and therefore inherently free of such dependencies. The occasional processing instruction is specially delimited so it can be found and converted for interchange, or when a different application renders the instruction irrelevant.

   References to external parts of a document are indirect. The mappings to real system storage are made in processing instructions that can be collected at the start of the document, where they can easily be modified for interchange.

   Delimiters can be changed with markup declarations to accommodate any reserved system characters.

4. There must be no national language bias.

   The character set used for names can be augmented by any special national characters. Generic identifiers, attribute names, and other names used in descriptive markup are defined by the user in structure, attribute, and entity declarations.

   The declaration names and keywords used in markup declarations can also be changed.

5. The language must accommodate familiar typewriter and word processor conventions.

   The "self-delimiting entity reference" and "data tag" features support typewriter text entry conventions.

6. The language must not depend on a particular data stream or physical file organization.

   The markup language has a virtual storage model in which documents consist of one or more storage entities, each of which is a sequence of characters. All real file access is handled by the processing system, which can decide whether the character sequence should be viewed as continuous, or whether it should reflect physical record boundaries.

7. "Marked up" text must coexist with other data.

   A processing system can allow text that conforms to the Standard to occur in a data stream of material that is not defined by the Standard, as long as the system can locate the start of the conforming text.

As the source file is scanned for markup and the various elements are recognized, the processing system executes the procedures associated with each element and attribute for that application. For other applications, different procedures can be associated with the same elements and attributes without changing the document markup.

When a processing instruction must be entered directly in a document, it is delimited differently from descriptive markup so that it can easily be located and changed for different applications.

2. Markup is formally defined for each type of document.

   A generalized markup language formalizes document markup by incorporating "document type definitions." Type definitions include a specification (like a formal grammar) of which elements and attributes can occur in a document and in what order. With this information it is possible to determine whether the markup for an individual document is correct (i.e., complies with the type definition) and also to supply markup that is missing, because it can be implied unambiguously from other markup that is present.

   Note: A more detailed introduction to the concepts of generic coding and the Standard Generalized Markup Language can be found in "Annex A. Introduction to Generalized Markup" on page 81. It should be read before reading the body of the Standard.

0.2 Objectives of the Standard

The Standard Generalized Markup Language standardizes the application of the generic coding and generalized markup concepts. It provides a coherent and unambiguous syntax for describing whatever a user chooses to identify within a document. The language includes:

- A syntax for descriptive markup of document elements. The specific vocabulary of generic identifiers and attributes, however, is defined by the user for different document types.

- Provision for arbitrary data content. In generalized markup, "data" is anything that is not defined by the markup language. This can include specialized "data content notations" that require interpretation different from general text: formulas, images, non-Roman alphabets, previously formatted text, graphics, etc.

- Entity references: a non-system-specific technique for referring to separate physical entities (files, et al.) in which parts of the document are stored.

- Markup declarations that customize the markup language. Declarations allow the user to specify the delimiters, generic identifiers, attributes, and other aspects of the markup used in a particular document.

- Special delimiters for processing instructions to distinguish them from descriptive markup. Processing instructions can be entered as needed for situations that cannot be handled by the procedures, but they can easily be found and modified when a document is sent to a different processing system.

For a generalized markup language to be an acceptable standard, however, requires more than just providing the required functional capabilities. The language must in fact be a metalanguage, in order to satisfy the constraints imposed by the need to use it in a multiplicity of environments. The major constraints, and the means by which the Standard Generalized Markup Language addresses them, can be summarized as follows:

1 Scope and Field of Application

1.1 Scope

This Standard:

1. Specifies a notation, or "document markup metalanguage," known as the Standard Generalized Markup Language. The language expresses the description of a document's structure and other attributes, as well as other information that makes the markup interpretable.

2. Defines classes of conforming document in terms of their use of components of the language.

3. Defines classes of conforming system in terms of their ability to process conforming documents.

4. Specifies how data not defined by the Standard (e.g., images, graphics, formatted text, et al.) can be included in a conforming document.

The Standard does not:

1. Identify or specify "standard" document types or text structures.

2. Specify the implementation of conforming systems and therefore does not define "conforming implementations."

3. Specify how conforming documents are to be created.

4. Specify the "data stream," "message handling system," file structure, or other physical representation in which conforming documents are stored or interchanged.

5. Specify the data content representation or notation for images, graphics, formatted text, et al., that are included in a conforming document.

1.2 Field of Application

The Standard Generalized Markup Language operates within the application layer (Level 7) of the Open Systems Interconnection architecture and and can be used for documents that are processed by any text processing or word processing system. It is particularly applicable to:

1. Documents that are interchanged among systems with differing text processing languages.

2. Documents that are processed by more than one application, even when the applications use the same text processing language.

Documents that exist solely in final formatted form are not within the field of application of this Standard.

Similarly, a system can allow data content not defined by the Standard to occur within a conforming document. The occurrence of such data is indicated by markup declarations to facilitate interchange.

8. The markup must be usable by both humans and programs.

   The Standard Generalized Markup Language is intended as a suitable interface for keyboarding and interchange without preprocessors. It allows extensive tailoring to accommodate user preferences in text entry conventions and the requirements of a variety of keyboards and displays.

   However, it is recognized that many implementers will want to take advantage of the language's information capture capabilities to provide intelligent editing or to create Standard Generalized Markup Language documents from a word processing front-end environment. The Standard accommodates such uses by allowing nongraphic bit combinations to be used as delimiters and by allowing binary fields to occur in text.

b. whether text characters can occur;

c. whether noncharacter data can occur.

A type definition does *not* specify:

1. The delimiters that are used to indicate markup.
2. Anything about the ways in which the document can be formatted or otherwise processed.

2.1.3 Procedures

Markup tags describe a document's structure of elements; they do not say how to process that structure. Many kinds of processing are possible, one of which is to *format* the text.

Formatting can be thought of as mapping the element structure onto paper or a display screen with graphic arts conventions. For example, the element "paragraph" could be displayed by setting the text of the element off from surrounding text with blank lines.

Processing is handled by *procedures*, which are written in the language of a formatter or other processing system. When a document is processed, a procedure is associated with each generic identifier (i.e., with each type of element). The procedure then processes the content of the element. In the case of formatting, for example, the procedure performs the actions that render an element into printed text or another display form.

Thus, production of a document begins when a user creates the text, marking it up as a particular document type. One of the facilities that can process that document is a formatter, which could have more than one set of procedures available.

For example, a document called "mybook," marked up as a "TechManual" document type, could be formatted in a number of ways by using a different *procedure set* each time. One set could produce output in single column CRT display style, another set in two column printed report style, and a third set in still another style.

When developing a completely new text application, then, a designer would create document type definitions, using the Standard Generalized Markup Language. Probably, he would also implement one or more procedure sets, using the languages of the systems that are to process the documents.

2.2 Markup

Markup is information that is added to the data of a document as an aid to processing it. In the Standard Generalized Markup Language, the markup in a document falls into four categories:

1. Descriptive Markup ("Tags")

   Tags are the most frequent and the most important kind of markup. They define the structure of the document, as described above.

2. Entity Reference

   Within a system, a single document can be stored in several parts, each in a separate unit of system storage, called an *entity*. (Depending on the system, an entity could be a file, a data set, a variable, a library member, etc.)

Separate entities are connected by *entity references* that can occur in a document's markup. An entity reference is a request for text—the entity—to be imbedded in the document at the point of the reference. The entity could have been defined either earlier within the document or externally.

The entity reference capability includes the functions commonly called *symbol substitution* and *file imbedding.*

3. Markup Declaration

   Declarations are statements that control how the markup is interpreted. They can be used to create and modify type definitions, to change delimiters, and to define entities.

4. Processing Instructions

   These are instructions to the processing system, in its own language, to take some specific action. Unlike the other kinds of markup, processing instructions are system-dependent, and are usually application-dependent as well. They normally need to be changed if the document is processed differently (e.g., formatted in a different style), or on a different system.

An implementation of the Standard Generalized Markup Language must recognize these four kinds of markup and handle them properly: that is, it must have a "markup parser."

> **Note: The Standard does not require that the markup parser be a dedicated program.**
>
> Part or all of the markup parsing could be done, for example, by a general parsing service function or by the procedures. The markup parser should be thought of solely as a *process*, with no implication of how it is implemented.

Markup occurs in a document according to a rigid set of rules. Some of the rules are dictated by the Standard Generalized Markup Language; they apply to all types of document. Those rules are called the *markup language syntax.*

Other rules are defined by the type definition for the type of document being processed.

Employing the markup language syntax, the markup parser must:

1. Scan the content of each element's text characters to distinguish the four kinds of markup from one another and from the text. (Noncharacter content data is not scanned by the parser.)

2. Replace entity references by their entities.

3. Interpret the markup declarations.

4. Give control to the processing system to execute processing instructions.

5. Interpret the descriptive markup tags to recognize the generic identifiers ("GIs") and attributes, and, following the rules of the document type:

   a. Determine whether each GI and its attributes are valid.

   b. Track the location in the document structure.

6. Give control to the processing system to execute the procedures associated with the GI and attributes.

Note:

The statement that the markup parser "gives control" to the processing system means only that the ensuing processing is not defined by this Standard. As was indicated earlier, there is no requirement for separate programs.

2.3 Distinguishing Markup from Text

The markup discussed in this section applies to all document types. The delimiter characters used are those recommended in the Standard. They can be changed in the document, in a manner that will be described later.

2.3.1 Descriptive Markup Tags

Descriptive markup tags identify the start and end of elements. There are four special characters that are important (see Figure 1):

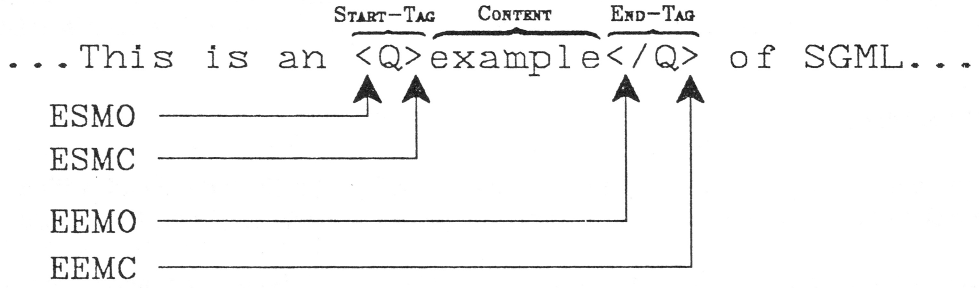


Figure 1. Element Markup

ESMO Element Start, Markup Open

This is a delimiter that indicates the start of an element. In the figure, "<" is the ***esmo***.

ESMC Element Start, Markup Close

The string from the ***esmo*** to the ***esmc*** is called the *start-tag*. In it the generic identifier ("GI") and all the attributes are given. In the figure, "Q" was the GI and ">" was used for the ***esmc***. (For subvocalization, pronounce the "mc" suffix as "mick.")

EEMO Element End, Markup Open

Between the ***esmc*** and the ***eemo*** is the *content* of the element, which can include text characters, subordinate elements, and noncharacter data. The ***eemo*** starts the *end-tag*, which generally contains a repetition of the GI to make the markup easier to read. In the figure, the two-character string "</" is the ***eemo***.

EEMC Element End, Markup Close

The ***eemc*** ends the ending markup and the whole element. In the figure, ">" was used again for the ***eemc***.

The scheme just described is the most general way of delimiting an element. The Standard Generalized Markup Language, though, allows a number of techniques of *markup minimization*. One simple technique is to omit the ***esmc*** (or ***eemc***, as the case may be) when a tag is immediately followed by another one.

For example, this markup:

```
<chapter><p>A short chapter.</p></chapter>
```

can be abbreviated to this:

```
<chapter<p>A short chapter.</p</chapter>
```

by omitting the ***esmc*** from the "chapter" start-tag and the ***eemc*** from the "p" end-tag.

Far more powerful minimization techniques, in which entire tags can be omitted, are available as optional features if a document type designer chooses to use them. They will be described later.

2.3.2 Other Markup

An entity reference begins with an "entity reference open" (***ero***) delimiter and ends with an "entity reference close" (***erc***). These are the ampersand and period, respectively, in the following example:

A markup declaration is delimited by "markup declaration open" (***mdo***, pronounced "mid-o") and "markup declaration close" (***mdc***, pronounced "mid-ick") delimiters, and a processing instruction by a "processing instruction open" (***pio***) and a "processing instruction close" (***pic***).

```
here is an imbedded <!markup declaration> example
and an imbedded <?processing instruction> example
```

To summarize:

Open	**Close**	**Type of Markup**
&	.	Entity Reference
<!	>	Markup Declaration
<?	>	Processing Instruction

2.3.3 Record Boundaries

A document could span many records of its storage entity. The record boundaries are known as the "record start" (***RS***) and "record end" (***RE***). An optional feature of the Standard Generalized Markup Language allows the record boundaries to be used as delimiter characters (discussed later). If a boundary is not a delimiter, its treatment depends on whether it occurs in text.

2.3.3.1 Record Boundaries in Text

When the content of an element includes text characters (i.e., instead of only tagged subelements or noncharacter data), the record boundaries are part of the text and might be significant to a processor of the document. A formatter, for example, normally interprets a record end as a space and ignores a record start.

However, record boundaries are ignored when they are caused by markup. That is,

- Record boundaries are ignored at the start or end of the content. For example,

  ```
  <p>
  Short paragraph text.
  </p>
  ```

  is the same as

  ```
  <p>Short paragraph text.</p>
  ```

- Record boundaries are ignored after a processing instruction or markup declaration. As a result,

  ```
  <p>
  Starting text
  <?instruction 1>
  <?instruction 2>
  ending text.
  </p>
  ```

  and

  ```
  <p>Starting text
  <?instruction 1><?instruction 2>ending text.</p>
  ```

  are equivalent. In other words, as far as the flow of text is concerned, declarations and processing instructions are simply ignored.

2.3.3.2 Other Record Boundaries

A document type could restrict the content of an element to tagged subelements; for example, list items within a list. Such an element could never contain text, even though its subelements might. Therefore, record boundaries between the subelements are not text, and are ignored. (The same is true for spaces and totally null records.)

Record ends in descriptive tags or declarations are treated as spaces. They serve as internal separators within the markup. Record starts are ignored.

The meaning of record boundaries within processing instructions depends on the processing system.

2.4 Document Structure

Descriptive tags serve two purposes:

1. They show the structural relationships among the elements of the document.

2. They identify each element's generic identifier (GI) and attributes.

The rules for specifying structure and attributes are defined by the Standard Generalized Markup Language for all documents. (Some rules were introduced in "Descriptive Markup Tags" on page 9; the others will be discussed later.) The *particular* elements and attributes allowed in a document, though, are defined by that document's type definition.

2.4.1 Type Definitions

A generic identifier (GI) identifies an element as a member of a class, or "type." A *type definition* is a set of markup declarations that apply to all documents (or elements) of a particular type.

Three kinds of declaration can occur in a type definition:

1. A *structure declaration* (**STRUC**) defines the GIs that can occur in each element, and in what order.
2. An *attribute declaration* (**ATT**) defines the attributes that can be specified for each element, and their possible values.
3. An *entity declaration* (**ENTITY**) defines the entities that can be referred to in documents of this type. For example, entity references can simplify the keying of frequently used lengthy phrases:

```
<!ENTITY SGML "Standard Generalized Markup Language">
```

To avoid repetitive keying, the type definition is usually stored as a separate entity. It is then incorporated in each document by means of an entity reference.

2.4.2 Structure Declarations

Elements can occur in a document only according to the rules of the document type definition. For example, unless the definition allows a paragraph inside a figure, it would be an error to put one there. The **STRUC** declaration is used to define these rules.

For each element in the document, the application designer specifies a **STRUC** declaration containing two parameters: the element's GI and a *model* of its content. The model parameter defines which subelements, text characters, and noncharacter data can occur in the content.

A model is made up of GIs, delimiters called *connectors* that put the GIs in order, and other delimiters, called *occurrence indicators*, that show how many times each GI can occur.

2.4.2.1 Connectors and Occurrence Indicators

For example, the structure of a textbook might look like this:

```
<!STRUC  textbook  (front, body, rear) >
```

Here, "textbook" is the GI whose structure is being defined, and "(front, body, rear)" is the model that defines it.

This example says that a textbook contains the GIs "front," "body," and "rear." (The GIs probably stand for "front matter," "body matter," and "rear matter," but this is of interest only to humans, not to the markup parser.)

A connector is used *between* GIs, to show how they are connected. This declaration uses the "sequence" connector (***seq***), the comma. It means that the elements must follow one another in the document in the same sequence as their GIs occur in the model.

An occurrence indicator is used *after* the GI to which it applies. There are none in this example: each element must therefore occur once and only once. To make the front matter and rear matter optional, the question mark, which is the optional occurrence indicator (***opt***), would be used.

```
<!STRUC  textbook  (front?, body, rear?) >
```

So far only the top level of a textbook has been defined. The type definition must also have structure declarations for "front," "body," and "rear," then for the elements contained in "front," "body," and "rear," and so on down to the text characters.

For the purposes of a simple example, the body can be assumed to be just a sequence of many paragraphs. That is, the only GI allowed in the body is "p" (for paragraph), but it can occur many times. To indicate multiple paragraphs, the GI is followed by a plus sign, which is the "required and repeatable" occurrence indicator (***rep***).

```
<!STRUC body (p+) >
```

The plus sign indicates that there must be at least one paragraph in the body. If, for some reason, it were desirable to allow a body to have no paragraphs at all, the added symbol would be an asterisk, which is the "optional and repeatable" occurrence indicator (***orep***).

```
<!STRUC body (p*) >
```

2.4.2.2 Groups

Suppose a textbook could have examples in the body, as well as paragraphs. If the GI for an example were "xmp," "many paragraphs or examples" would be indicated like this:

```
<!STRUC body (p | xmp)+ >
```

The preceding example introduces two new pieces of notation. The first is the use of the vertical bar as the "or" connector (***or***). The expression "p | xmp" means "either a paragraph or an example."

The other new notation is the use of parentheses as "group open" (***grpo***) and "group close" (***grpc***) delimiters. The parentheses group an expression into a single *model-token* to which an occurrence indicator can apply. The grouping is needed because the occurrence indicators ("?," "+," and "*") have higher precedence than the connectors ("," and "|").

Therefore, it would not be effective to say

```
<!STRUC body (p+ | xmp+) >
```

because it would mean "either many paragraphs or many examples" rather than the desired "many intermixed paragraphs or examples."

There is one more kind of connector to consider. Suppose the front matter of the textbook had a title page that contained a title, an author's name, and a publisher, but in *any* order. The ***seq*** connector cannot be used because that requires a specific order. Nor can the ***or*** connector, because that chooses only one of the three elements.

Instead, the "at" sign ("@"), which is the "and" connector (***and***) should be used. It indicates that all of the GIs in the group must occur, but in any order.

```
<!STRUC titlepage (title @ author @ publisher) >
```

By now it should be evident that a model is also a group, or a group with an occurrence indicator tacked onto it. In other words, a model must always start with a ***grpo*** delimiter.

There is yet another use for groups. The body matter and rear matter might have the same structure declaration.

```
<!STRUC body (p | xmp)+ >
<!STRUC rear (p | xmp)+ >
```

Keying effort could be saved and the similarity emphasized by letting the declaration apply to a group of elements.

```
<!STRUC (body | rear) (p | xmp)+ >
```

2.4.2.3 Using Entity References

Suppose two elements have almost the same content model, but not quite.

```
<!STRUC body (p | xmp | h1 | h2 | h3 | h4)+ >
<!STRUC rear (p |       h1 | h2 | h3 | h4)+ >
```

Here, the body and rear matter both have paragraphs and headings, but only the body can have examples.

Repetitive parts of markup declaration parameters can be handled with entity references, just like repetitive text in a document. The only difference is that the names of entities used primarily in declaration parameters ("parameter entities") begin with a special character, the number sign ("#").

```
<!ENTITY #h1to4 "h1 | h2 | h3 | h4"    >
<!STRUC  body   (p | xmp | &#h1to4.)+ >
<!STRUC  rear   (p |       &#h1to4.)+ >
```

2.4.2.4 Text Characters

Up to now, all of the elements discussed contained only other elements. Eventually, though, an application must deal with the actual text, that is, the point where there are no more tags.

Text can be referred to with the entity reference "&#CHAR.," which means "any usable character." It is a special parameter entity that is provided automatically by the markup parser. The declaration

```
<!STRUC p (&#CHAR.+) >
```

says that a paragraph is a string of one or more characters (including record boundaries and spaces).

Incidentally, the reason "&#CHAR." is in uppercase is simply as a reminder that it is defined by the markup parser. Normally, the markup parser ignores the case of all names, including GIs and entities.

In most documents, elements that contain text characters can also contain other tagged elements. Such elements normally include short quotations, footnote and figure references, various types of highlighted phrases, and specialized references or citations defined for a particular document type.

For example, the structure of a paragraph might be defined as:

```
<!ENTITY #phrase "(quote | citation | reference)" >
<!STRUC  p          (&#CHAR. | &#phrase.)*        >
```

These declarations indicate that a paragraph contains characters mixed with "phrase" elements. There can be none or many of them.

A more complex definition could indicate that a paragraph had to start with a character:

```
<!STRUC p (&#CHAR.,(&#CHAR. | &#phrase.)*) >
```

This declaration calls for "a character, followed by either characters or 'phrase' tags, repeatedly."

The previous definition could be extended to allow empty paragraphs:

```
<!STRUC p (&#CHAR.,(&#CHAR. | &#phrase.)*)? >
```

In other words, if the outermost group is made optional, the paragraph can be empty. An alternative model that allows empty paragraphs could be as follows:

```
<!STRUC p (&#CHAR.?,(&#CHAR. | &#phrase.)*) >
```

but it would not meet the objective of requiring a character to be the first thing in a paragraph. That model allows the starting text to be omitted and thus permits a "phrase" to occur first, which is just a more complicated way of saying

```
<!STRUC  p (&#CHAR. | &#phrase.)* >
```

2.4.2.5 Data Content

If the body of the textbook contained photographs, the declarations could be

```
<!ENTITY #h1to4 "h1 | h2 | h3 | h4"  >
<!STRUC  body  (p | xmp | &#h1to4. | photo)+ >
```

A photo is typically represented by a string of bits that stand for the colors of the various points in the picture. These bit combinations do not stand for characters as they do in text and in markup; they have to be interpreted as a unique notation that is not defined by the Standard Generalized Markup Language.

To show that an element contains an unknown *data content notation*, a parser-declared entity reference can be used:

```
<!STRUC photo (&#BITDATA.) >
```

The markup parser does not scan bit data content. The procedure is responsible for locating the end of the data.

Data content can be intermixed with text characters and/or elements:

```
<!STRUC figbody (&#BITDATA. | &#CHAR.)+ >
```

Whenever unparsable data content could occur, the markup processor stops scanning and the procedure takes over. The procedure is responsible for processing data content that is actually present, and returning control to the markup parser whenever a data content portion ends.

2.4.2.6 Summary of Model Delimiters

The following list summarizes the delimiters used in models and the character strings that the Standard recommends for them:

Grouping:

String	Name	Meaning
(	GRPO	Opens a group. The expression within the group is treated as a unit for other operations.
)	GRPC	Closes a group.

Occurrence Indicators:

String	Name	Meaning
?	OPT	Optional: can occur 0 or 1 time.
+	REP	Required and repeatable: can occur 1 or more times.
*	OREP	Optional and repeatable: can occur 0 or more times.

Connectors:

String	Name	Meaning
,	SEQ	All of the connected elements must occur in the document in the same sequence as in the model.
|	OR	One of the connected elements must occur, but not all.
@	AND	All of the connect elements must occur in the document, but in any order.

2.5 Attributes

A descriptive tag normally includes the element's generic identifier (GI) and may include other attributes as well. The GI is normally a noun; the attributes are nouns or adjectives that describe significant characteristics of the GI. (They should not be verbs, which are procedural rather than descriptive, nor should they be formatting parameters.)

The *particular* attributes allowed for a given element are defined by the type definition, which also determines the range of values that an attribute can have and what the default values are.

2.5.1 Specifying Attributes

A sample tag with two attributes is shown in Figure 2 on page 17.

The attributes follow the GI in the descriptive tag. Each attribute is specified by giving its *name*, the value indicator delimiter (***vi***), and the attribute *value*. In the example, the attribute named "security" was given the value "internal use," and the attribute "id" was given the value "test."

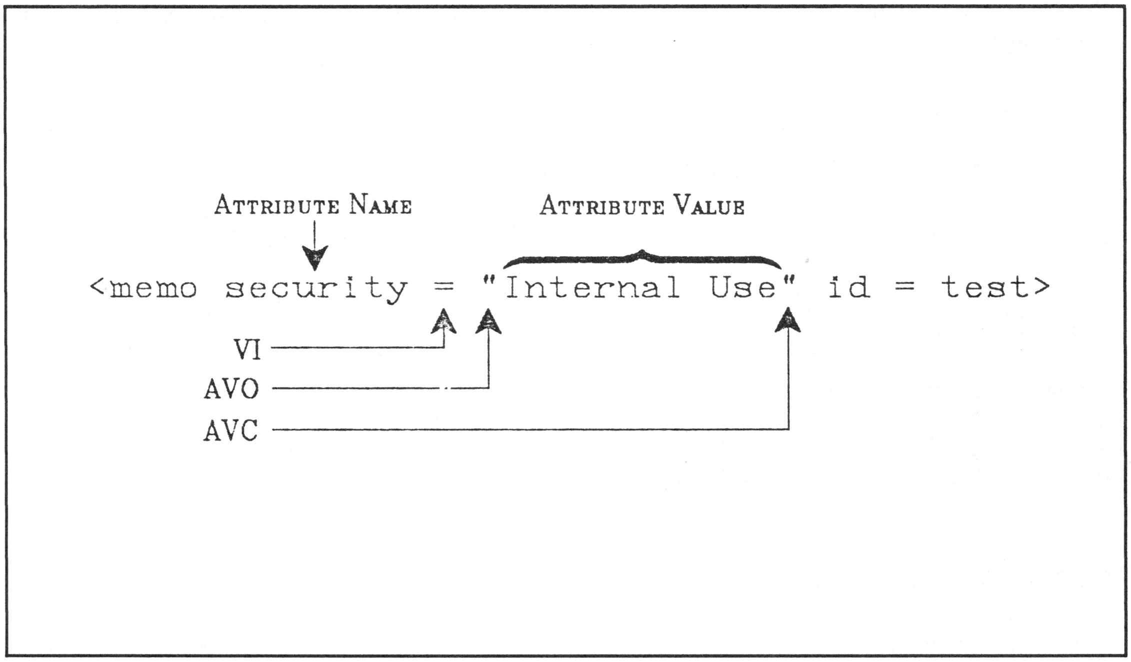


Figure 2. **Start-tag with 2 Attributes**

2.5.1.1 Names

Attribute names, such as "security" and "id" in the example, share certain properties with other kinds of markup language names already encountered, such as entity names and GIs. A name must consist only of characters that have been designated as "name characters" (***NAMECHAR***), and it must start with one of a subset of the name characters called the "name starter" characters (***NAMESTRT***).

Normally, the name characters are the letters and numerals, and the name starter characters are the letters only. Lowercase letters in a name are normally treated as though they were uppercase, so it makes no difference in which case a name is keyed.

A name normally has a maximum length of eight characters, but the limit can be changed in a manner that will be explained later.

2.5.1.2 Attribute Values

The value of an attribute is syntactically similar to the content of an element. It cannot, however, contain markup declarations or processing instructions. The value is bounded by delimiters called the "attribute value open" (***avo***) and the "attribute value close" (***avc***), both of which are normally quotation marks ("). Record boundaries preceding or following the value are ignored, unless they are interpretable as delimiters.

```
<h2 stitle="Attribute Value">
```

is equivalent to

```
<h2 stitle="
Attribute Value
">
```

If the attribute value is limited to name characters, the delimiters are not required (but are permitted):

```
<standard security=public>
```

A null attribute value is indicated by the ***avo*** and ***avc*** in succession:

```
<listing name=Jones phone="555-1234" altphone="">
```

The following list summarizes the delimiters and the default character string assignments recommended by the Standard:

String	**Name**	**Meaning**
=	**VI**	Value Indicator
"	**AVO**	Attribute Value Open
"	**AVC**	Attribute Value Close

2.5.2 Declaring Attributes

For every element, the type definition must include, in addition to the **STRUC** declaration, information that establishes the element's attributes. Logically, such information should be part of the **STRUC** declaration, but to keep the declarations simpler, it has been broken out into a separate declaration called an *attribute declaration* (**ATT**). If there is a **STRUC** declaration for an element, but no **ATT** declaration, then the element simply has no attributes.

2.5.2.1 Attribute Declaration Syntax

An attribute declaration begins with an "element" parameter that is the same as the one on the **STRUC** declaration. In particular, the element parameter can be a group if the same attributes apply to all elements in the group.

The declaration then includes, for each attribute:

1. the attribute name,

2. a model of its value, and

3. a default value that will be used if the attribute is not specified in the document.

The attribute name parameter can also be a group, as long as all attributes in the group have the same model and default value.

The attribute declaration has more parameters than the declarations shown up to now. When dealing with lengthy declarations, it is helpful to add explanatory comments to them. In a markup declaration, comments can occur between any parameters. They begin and end with two hyphens, the "parameter separator open" and "parameter separator close" delimiters (***pso*** and ***psc***). One use for comments is to put a heading over the parameters:

```
<!ATT --
NAME       VALUE                   DEFAULT -- >
```

The **ATT** declaration requires that some of the parameters be repeated for each attribute of the element. As declarations can span record boundaries, a tabular form of entry can be used if desired. The following **ATT** declaration for the memo element in Figure 2 on page 17 is in tabular form:

```
<!ATT memo --
NAME       VALUE                DEFAULT --
status     ("final"|"draft")    "final"
security   (&#CHAR.+)            REQUIRED
version    (&#NUM.)              "01"
sender     (&#NAME.)             UNKNOWN
>
```

The meaning of the declaration is as follows:

status	The "status" attribute must have a value of "final" or "draft." As the attribute was not specified in Figure 2, the parser will act as though the default value of "final" was entered.
security	The "security" attribute has a string of one or more characters as its value. The keyword "REQUIRED" in the default parameter indicates that the attribute must always be specified in the document.
version	The "&#NUM." in the model of the "version" attribute is a reference to a parameter entity, declared by the parser, with the meaning "a decimal number with 1 or more digits." The default value is the decimal number "01."
sender	The "sender" attribute must have a syntactically valid Standard Generalized Markup Language name as a value. The keyword "UNKNOWN" indicates that the attribute is optional, but that there is no known default that the parser can supply if the attribute is not specified in the document (which means the procedures will assume some default value on their own).

2.5.2.2 Complex Attribute Values

An attribute value might logically consist of a number of elements. For example, an illustration of output displayed by a computer could have two colors associated with it: green and black. As an attribute can be specified only once in a tag,

```
<display color=black color=green>
```

could not be entered, as it would be an error.

Instead, declaring that the "color" attribute can be specified as a list would allow both colors to be specified in a single attribute value:

```
<!ATT display color (&#NAMES.) UNKNOWN >
<!>
<display color="black green">
```

The "&#NAMES." is a reference to a parameter entity, declared by the parser, that means "a list of one or more Standard Generalized Markup Language names, separated by spaces, record boundaries, or null records."

The "<!>" is simply a null markup declaration used as a separator in the source document. It is treated as a comment and ignored.

A complex attribute could be avoided altogether in this case by defining two separate attributes:

```
<!ATT display (bgcolor | fgcolor) (&#NAME.) UNKNOWN >
<!>
<display bgcolor=black fgcolor=green>
```

Attribute value models in general are quite limited in comparison to structure content models. With the optional attribute extension feature (discussed later), though, a designer could choose to allow much greater attribute value complexity if it were necessary.

2.5.2.3 Changing Default Values

A short form of the **ATT** declaration can be used to change the default value of a previously defined attribute. In this case, only the element, attribute name, and default parameters are specified.

```
<!ATT memo status "draft">
```

The short form of the **ATT** declaration is used mostly within a document rather than in a type description.

2.6 Entities

Entity references have figured prominently in some of the declaration examples that have already appeared. Although an entity reference has a superficial resemblance to a programming language variable, it is actually a symbolic reference to a portion of the document. Entity references permit a number of useful techniques:

1. A short name can be used to refer to a lengthy or unkeyable text string.
2. Parts of the document that are stored in separate system files can be imbedded.
3. Documents can be exchanged among different systems more easily because references to system-specific objects (such as unkeyable characters) can be in the form of entity references that are resolved by the receiving system.
4. The result of a dynamically executed processing instruction (such as an instruction to retrieve the current time of day) can be imbedded as part of the document.

2.6.1 Entity Reference Syntax

An entity can be referred to anywhere that text characters can be entered: that is, in the content of elements and delimited attribute values, and in markup declaration parameters. (The ability to do so in processing instructions depends on the processing system.)

An entity reference is a name delimited by an "entity reference open" (***ero***), normally an ampersand, and an "entity reference close" (***erc***), normally a period:

```
printed at &site. on
```

If an entity reference is followed by a space or record end, the ***erc*** can be omitted:

```
printed at &site on
```

2.6.2 Declaring Entities

Before an entity can be referred to it must be declared with an entity declaration (**ENTITY**) The declaration

```
<!ENTITY uta "United Typothetae of America">
```

means that entering "&uta." in the text will be the equivalent of entering "United Typothetae of America."

The entity can be canceled by a declaration of the form,

```
<!ENTITY uta CANCEL>
```

which means that a reference to "&uta." will be considered an error.

An entity does not inherently begin with a record start or end with a record end. If record boundaries are wanted around an entity, they should be put around the entity reference. The source

```
<p>The &uta. is a printing organization.</p>
```

resolves to

```
<p>The United Typothetae of America is a printing organization.</p>
```

while the source

```
<p>Printing organizations:
&uta.
Society of Scientific, Technical, and Medical Publishers
</p>
```

resolves to

```
<p>Printing organizations:
United Typothetae of America
Society of Scientific, Technical, and Medical Publishers
</p>
```

2.6.2.1 Literals

In the declaration

```
<!ENTITY finis "-|-" >
```

the quoted string is a *character literal*; the quotation marks are the "character literal open" (***clo***) and "character literal close" (***clc***) delimiters. The **ENTITY** declarations illustrated so far all used character literals to specify the entity.

There are, however, other ways to specify a character string besides using a character literal. Each text character occupies a position in the document's *coded character set*. The total number of positions depends on the size of the *code*, that is, on the number of binary digits ("bits") used to represent each character.

For example, the character set known as ISO 646 International Reference Version (ISO 646 IRV) is a 7-bit code. There are 128 *bit combinations* possible with 7 bits, ranging from the values 0 through 127 in the decimal number base (7F in hexadecimal; 177 in octal). Bit combination 45 (decimal) is assigned to the hyphen. As the markup parser deals exclusively with bit combinations, not characters, a character literal is really just a way of saying "the bit combination assigned to this character."

A *numeric literal* can be used in any of the three number bases to enter a bit combination directly. The four declarations in the following example are equivalent in any character set which assigns the hyphen to position 45:

```
<!ENTITY hyphen "-"    >
<!ENTITY hyphen D/45   >
<!ENTITY hyphen H/2D   >
<!ENTITY hyphen O/055  >
```

The numeric literals all begin with a number base identifier delimiter (***dec***, ***hex***, or ***oct***) followed by a "numeric literal divider" (***nld***), the stroke. The hexadecimal literal then has 2 characters per number and the octal literal has 3. The decimal has whatever number of characters per number is required, with additional ***nld*** delimiters separating the numbers.

The following declarations are equivalent in ISO 646 IRV:

```
<!ENTITY finis "-|-"        >
<!ENTITY finis D/45/124/45  >
<!ENTITY finis H/2D7C2D     >
<!ENTITY finis O/055174055  >
```

Different types of literal can be concatenated by grouping them with a ***seq*** connector:

```
<!ENTITY finis ("-", D/124, H/2D) >
```

A numeric literal is useful for storing bit combinations that cannot be keyed directly:

```
<!ENTITY bullet D/4 >
```

The following list summarizes the delimiters used in literals and the character strings that the Standard recommends for them:

String	Name	Meaning
"	CLO	Character Literal Open
"	CLC	Character Literal Close
D	DEC	Decimal numeric literal.
H	HEX	Hexadecimal numeric literal.
O	OCT	Octal numeric literal.
/	NLD	Numeric Literal Divider.

2.6.2.2 Processing Instructions

A processing instruction can be stored as an entity. It will be ignored when the entity is created but executed when a reference to the entity occurs.

```
<!ENTITY part2 PI ><?IMBED part2.textfile>
```

The keyword "PI" indicates that the processing instruction immediately following the **ENTITY**declaration should be stored as the entity.

If "IMBED" were a processing instruction that retrieved the contents of a file and "part2.textfile" a system file identifier, a reference to "&part2." would cause the contents of "part2.textfile" to be included in the document at the point of the reference.

2.6.2.3 Entities with Entity References

Another entity can be referred to in the literal.

```
<!ENTITY prefix "&bullet.  " >
```

It will be resolved when a reference is made to the entity. It is not resolved during storage of the entity because no entity is parsed during storage, other than to find its end.

2.6.2.4 External Entities

In many text processing systems, there are multiple classes of storage, such as files, library members, macro definitions, and symbols for text strings. Such system dependencies can be kept out of the body of the document by referencing external storage objects as entities:

```
<!ENTITY part2 EXTERNAL>
```

If the entity name is not sufficient to enable the system to identify the storage object, additional information can be specified in an associated executable processing instruction:

```
<!ENTITY part2 EXTERNAL XPI><?mapfile part2 user.part2.textfile>
```

The "XPI" parameter indicates that an executable processing instruction ("the XPI") follows the entity.

Some processing systems might also require an executable processing instruction when the entity is other than an external object, for example, in order to provide information about how the entity is to be stored. In all cases, the system dependencies are confined to the "XPI" instruction, where they can easily be located and changed when the document is sent to another system.

2.6.3 Entity Sets

A set of entities might be defined for use in all documents of a given type. The declarations for such an *entity set* should be included in the document type definition, along with the structure and attribute declarations.

Entity sets can also be defined for graphic symbols and terminology of specialized subject areas, such as mathematics or chemistry. The entity declarations for a such a set should be stored as an entity; they can then be incorporated in any document (regardless of document type) that discusses the subject area.

2.6.4 Entities in Markup Declarations

Entities used in text are often created for an individual document by its author or the person who keys it. Markup declarations, though, are usually created by support personnel for a class of documents. In order to avoid naming conflicts, names of entities used in declaration parameters must all begin with a character from the "parameter entity starter" (***PENTSTRT***) portion of the character set, rather than ***NAMESTRT***.

Parameter entities can also be used within the text of the document, but normal text entities cannot be used in a declaration.

2.7 Marked Sections

A *marked section* is a portion of a source document that has been labeled for a special purpose, such as storing it as an entity or ignoring it during certain processing runs.

2.7.1 Ignoring a Marked Section

If a document is processed on two different systems, a processing instruction that applies to one will not be understood by the other. The markup parser can be made to ignore one of the commands if the section of the document that contains it is marked as a section to be ignored:

```
<!MS IGNORE><?instruction for System A><!MSE>
```

The marked section in the example consists of three parts: the *marked section start*, which has the parameter "IGNORE,"

```
<!MS IGNORE>
```

the content,

```
<?instruction for System A>
```

and the *marked section end*.

```
<!MSE>
```

The other processing instruction would be marked as one to be included; that is, it would not have the "IGNORE" parameter:

```
<!MS><?instruction for System B><!MSE>
```

Sending the document to system A requires exchanging the marked section starts for the two sections. This can be done most easily by using two entities, one for each processing system. One entity would contain the character string "IGNORE" and the other would contain a blank:

```
<!ENTITY systema "IGNORE" >
<!ENTITY systemb " "      >
```

The "IGNORE" keyword would not be used directly in any marked section start. Instead, there would be a reference to one of the two system-dependent entities. Given the previous declarations, the instruction for "System A" in the following example will be ignored, while the one for "System B" will be executed:

```
<!MS &systema.><?instruction for System A><!MSE>
<!MS &systemb.><?instruction for System B><!MSE>
```

Every other marked section in the document that refers to "&systema." or "&systemb" will be affected similarly.

Now, if the two **ENTITY** declarations are reversed, as follows,

```
<!ENTITY systema " "       >
<!ENTITY systemb "IGNORE" >
```

every marked section in the document that refers to "&systemb." will be ignored.

Note that, as a marked section is more likely to be created by the end user of a document, normal entity references can be used instead of parameter entity references.

2.7.2 Unparsable Sections

A marked section can be labeled as one that is not to be parsed:

```
<!MS DATA>
<?instruction>
<p>A paragraph with an &entityx.
reference.</p>
<?instruction>
<!MSE>
```

The content of the section is treated as data content, so the two processing instructions and the entity reference will not be recognized or processed as such. It is possible to ignore the entity reference because a marked section end must occur in the same entity as the marked section start.

The only parsing done is recognition of nested marked section starts and ends, so the correct end will be used for the section. Unlike normal parsing, this scan includes processing instructions and data content, so those things must not have bit combinations that look like marked sections. Such bit combinations can be used in data by imbedding them with an entity reference.

2.7.3 Temporary Sections

A marked section can be flagged as temporary for easy identification and removal:

```
<!MS TEMP><?newpage><!MSE>
```

Such sections are useful for temporary "fixes" when a processor does not get things quite right.

The "IGNORE," "DATA," and "TEMP" parameters can be used concurrently.

2.7.4 Storing a Marked Section as an Entity

A marked section can be stored as an entity so it can be incorporated at many points in the document:

```
<!ENTITY phrase1 MS>
<!MS DATA>
a repeated phrase with
an &entity1. reference
<!MSE>
```

Given these declarations, the input

```
This is &phrase1. in it.
```

will resolve to

```
This is a repeated phrase with
an &entity1. reference in it.
```

The reference to “&entity1.” did not resolve because of the “DATA” parameter on the marked section start.

Note that there was no record start before the content of the marked section nor a record end after it. The following example would have produced them:

```
This is
&phrase1.
in it.
```

A marked section is not parsed or processed while it is being stored. It is, however, scanned for the ending (just as if it were being ignored).

2.7.5 Using Delimiter Characters as Data

The Standard Generalized Markup Language does not compel the use of particular characters (i.e., bit combinations) as delimiters. Instead, it defines delimiter *roles* and establishes recommended character string assignments for them. Although there are 41 such roles, the user’s ability to enter data characters freely is not seriously affected:

- Many of the delimiters occur only within markup declarations or tags, not in text.
- The same characters can be used for more than one role.
- Most delimiter roles are context-sensitive; only two (***net*** and ***nst***, which are explained later) are absolute, and they are not required.
- Multicharacter strings can be used as delimiters to minimize the likelihood of their occurring as data.

When a problem does arise, it can be solved by entering the character indirectly with a reference to an unparsable entity. Only three such entities are needed for the normal delimiters used in the Standard:

```
<!ENTITY amp   MS><!MS DATA>&<!MSE>
<!ENTITY ltsym MS><!MS DATA><<!MSE>
<!ENTITY sdq   MS><!MS DATA>"<!MSE>
```

The entity references “&amp.” and “&ltsym.” can now be used freely in text whenever the ampersand or less-than symbol are wanted; the "DATA" parameter assures that they will not be treated as delimiters.

The “&sdq.” is needed only when it is necessary to enter a straight double quotation mark as data in an attribute value, because this character has no significance as a delimiter in an element’s content.

2.8 Character Sets

The idea of a character set has already been discussed. It is a sequence of bit combinations (numbers, more or less) that are mapped to numerals, letters, and other characters. Many character sets have been defined, to accommodate a variety of national alphabets, scientific notations, keyboards, display devices, and processing systems.

The Standard Generalized Markup Language divides a document's character set into five mutually exclusive categories. The bit combinations assigned to each category can be changed in a manner that will be described later.

SYSCHAR	Bit combinations that never occur in parsable text (they can occur in unparsable data content) which the system can use for its own purposes, such as padding or delimiting. These are usually nonprinting control characters, such as the line feed and carriage return.
NAMECHAR	Bit combinations for characters that can be used in a name, which must include the uppercase and lowercase A through Z and the numerals 0 through 9.
SPACE	The bit combination for a space character.
PENTCHAR	The bit combinations that are allowed to begin a parameter entity name, normally the number sign (#).
BITCOM	All other bit combinations, such as punctuation and mathematical symbols. Delimiters are usually chosen from this category.

There is also a subset of ***NAMECHAR*** called ***NAMESTRT*** that contains bit combinations for characters that can start a name. It must include uppercase and lowercase A through Z. Each character in ***NAMECHAR*** maps to an associated uppercase form that is used for names. (Uppercase letters and numerals map to themselves.)

There are also three *pseudocharacters* that represent record boundary conditions: the record start and record end (***RS*** and ***RE***) that have already been introduced, and the null record (***NR***). They are called "pseudocharacters" because they are not represented by bit combinations in the normal way. If the optional record boundary recognition and delimiter modification features are used, the pseudocharacters can be used as delimiters.

2.9 More About Structure

2.9.1 Exceptions to the Model

The model specifies the elements that occur at the top level of the content (the subelements). However, there are instances when it may be necessary to refer to elements that are further down. An optional parameter of **STRUC**, called the "exception" parameter, is used for this purpose.

1. Included Elements

   In an indexed book, entry elements for the index could be interspersed anywhere in the source. During formatting, the procedures would collect, sort, and merge them, along with their page numbers, and print the index.

   It is cumbersome to express the relationship of the book to its index entries in the usual way, as the entries would have to be included in almost every model. Instead, the exception parameter can be employed:

   ```
   <!STRUC --
   ELEMENT  CONTENT                  EXCEPTION?  --
   textbook (front?, body, rear?)   (+entry)
   >
   ```

The plus sign is the ***with*** prefix delimiter. It means that index entries can occur anywhere in a textbook, even among the text characters.

(Incidentally, the occurrence indicator on the heading is just a reminder that the exception parameter is optional.)

2. Excluded Elements

   It might be desirable to keep some GI from showing up at *any* level of an element being defined. For example, figures could be kept from nesting with the following declarations:

```
<!STRUC --
ELEMENT   CONTENT                EXCEPTION?  --
fig       (figbody, figcap?)     (-fig)
figbody   (artwork | p+)
p         (&#CHAR.+ | fig )
>
```

   The model clearly does not allow a figure to occur at the top level of another figure, but the figure body could contain a paragraph that could contain a figure. The exception parameter with the ***less*** prefix (hyphen) prevents such difficulties.

It is possible to specify inclusion and/or exclusion for a group of GIs. Each must have its own prefix, and the ***or*** connector must be used between them.

```
<!STRUC --
ELEMENT   CONTENT                EXCEPTION?  --
fig       (figbody, figcap?)     (-fig | -xmp | +gloss) >
```

The following list summarizes the delimiters used in the exception parameter and the character strings that the Standard recommends for them:

String	**Name**	**Meaning**
-	**LESS**	Exclusion
+	**WITH**	Inclusion

2.9.2 Null Content

An element can be defined for which the user never enters the content, for example, a figure reference for which a procedure will always generate the content during processing. The **STRUC** declaration would look like this:

```
<!STRUC --
ELEMENT   CONTENT  EXCEPTION? --
figref    ()
>
```

2.9.3 Alternative Models

Some documents are formatted and printed as separate chapters. Separate formatting prevents automatic figure references across chapter boundaries, although they would still be possible within a chapter. A **STRUC** declaration could support the two types of figure reference by specifying that the content is null only under certain conditions (the intra-chapter case), and that at other times it will be entered explicitly (the cross-chapter case).

Alternative models for an element are declared by entering them successively within the same **STRUC** declaration. The first alternative must have a null content model and must identify, in the exception parameter, one or more attributes that must be present or absent in the document for the content to be null. The second alternative can have any model that is not null; its exception parameter (if any) is interpreted normally.

```
<!STRUC --
ELEMENT  CONTENT          EXCEPTION? --
figref   ()               (+refid)
figref   (&#CHAR.+)
>
```

The example declares that the content is null if the "refid" attribute is specified in the document. If the "refid" attribute is not specified, the content contains text characters. The following markup is therefore valid in the document:

```
Here is text with a generated figure reference to
<figref refid=figdavis> and a user-entered reference to
<figref>Figure A-1 in the Appendix</figref> as well.
```

2.10 Data Content

The data content of a document is the portion that is not markup. It has two major characteristics:

1. Its *representation* determines whether the markup parser can scan it.

   There are two main classes: character data, in which the bit combinations represent characters, and bit data, in which the bit combinations (singly or collectively) usually represent binary values. The parser can only scan character data that is in the document's character set, that is, the character set in which the markup is done.

2. Its *notation* determines how the character or bit strings will be interpreted by the procedures.

   In a natural language notation, for example, the character string "delta" would be interpreted as an English word, while in a scientific notation it would be interpreted as a single graphic, a Greek letter.

2.10.1 Data Content Representations

The Standard recognizes three data representations: character set data, undefined character set data, and bit data.

2.10.1.1 Character Set Data (CSDATA)

In character set data, each bit combination represents a character in the document's character set.

Before a character can be considered character set data, it normally must be parsed to determine whether it is data or markup. In a paragraph with the following structure, a character could either be data or part of a "phrase" or "quote" element tag:

```
<!STRUC p (&#CHAR. | phrase | quote)+ >
```

A character could also be part of a markup declaration, processing instruction, entity reference, or other markup that is allowed in the content of an element. Only if it is not a tag or other markup would it be treated as data and passed to the procedures for processing.

A character used for markup is not normally passed to the procedures, but when the optional "DATATAG" feature is used, a character could be both markup *and* data. A space, for example, could serve as the end-tag for a word and still be part of the data of the sentence in which the word occurs.

It is possible to enter character set data directly, without it being parsed in the usual way. An element's **STRUC** declaration can declare it to contain character set data by referencing a special parser-declared entity:

```
<!STRUC formula (&#CSDATA.) >
```

If an element contains declared character set data, it cannot contain anything else. The markup parser scans it only to locate the end-tag; other markup is ignored.

Attribute values can also be declared to contain character set data, and markup declaration comments always do. In all these cases, the data is only scanned to locate the closing delimiter.

2.10.1.2 Undefined Character Set Data (UCSDATA)

In undefined character set data, the bit combinations represent characters, but not in the document's character set. The data, for example, could be in a code that assigns multiple bit combinations to a single character. Such a scheme is used for languages that require more characters than there are bit combinations in the document's character set.

UCSDATA can be entered only when permitted in a declaration, where it is referenced by a parser-declared entity:

```
<!STRUC kanji (&#UCSDATA.) >
```

The UCSDATA is scanned by the application procedure, not the markup parser. If elements and UCSDATA are intermixed in an element's content, the content must be parsed by the application procedure, which must return control to the markup parser when an element (or other markup) is encountered.

2.10.1.3 Bit Data (BITDATA)

In bit data, the bit combinations do not represent characters.

BITDATA can be entered only when permitted in a declaration, where it is referenced by a parser-declared entity:

```
<!STRUC photo (&#BITDATA.) >
```

Bit data is treated in the same manner as UCSDATA; it is never scanned by the markup parser.

2.10.2 Data Content Notations

Potentially, each procedure could use a totally different data content notation. In practice, however, data content notations tend to fall into a few distinct classes.

2.10.2.1 Character Notations

Character notations are those that use CSDATA or UCSDATA representations. Some common classes of character notation are natural languages, scientific notations, and formatted text.

Natural Languages*:* When a natural language notation is used, the procedures interpret the characters (and record boundary pseudocharacters) as implicitly marked text elements. For example, a sequence of characters bounded by interword spaces could be recognized as a "word" element, a sequence of words terminated by suitable punctuation as a "phrase" or "sentence" element, and so on. In such notations, an ***RE*** is usually interpreted as an interword space, and an ***RS*** is usually ignored.

For additional flexibility, a natural language text notation can be supplemented by explicit markup for specialized variants of elements that are normally implicit (e.g., quoted phrases, programming keywords, emphasized phrases, etc.).

In conventional text processing systems, implicit markup conventions and the trade-offs between explicit and implicit markup are built in. In the Standard Generalized Markup Language, the data tag markup minimization feature can allow a user to specify the extent to which such elements will be recognized by the markup parser and the extent to which they will be interpreted by the application procedures.

Natural language notations occur in elements such as paragraphs, list items, and headings. The element content is usually characters or UCSDATA intermixed with elements.

Scientific Notations*:* These notations look like natural language notations, but the words and phrases are have meaning to the application. In a mathematical notation, for example, the phrase "3 over 4" could cause the fraction "3/4" to be printed.

Scientific notations occur in elements such as formulas, equations, chemical structures, formal grammars, musical themes, and so on. The element content is usually CSDATA, which at least allows systems that cannot interpret the notation to process the element as text characters.

Formatted Text*:* Formatted text notations are similar to natural language notations, and features of both might be incorporated in a single notation. Their purpose is to identify textual elements of the "result document" produced by a previous formatting application. A formatted text notation will include character sequences that identify formatted line endings, spaces and hyphens introduced for alignment purposes, font change codes, superscripts and subscripts, etc.

These notations are found in the same types of elements as natural language notations: paragraphs, headings, et al., usually when the optional coexistent structure feature is used. The element content is normally characters or UCSDATA intermixed with elements.

2.10.2.2 Bit Data Notations

Bit data notations are those that use the BITDATA representation. The bit combinations (singly or collectively) can represent binary values or a mixture of binary values and characters.

Binary Values*:* Binary values are interpreted as grey scale or color values for pixels in an image, digitized audio, musical wave forms, or as other sets of numeric or logical values. The principal application of binary value notations in text processing is in the representation of illustration elements: half-tones, photographs, etc.

Record boundaries are frequently ignored in binary notations, but need not be.

Mixed Binary and Character: Scientific and formatted text data notations can be designed as a mixture of characters and binary fields. Such notations are treated as unparsable bit data notations because markup delimiter bit combinations could occur accidentally in the binary values. As the application procedures can understand the context in which each bit combination occurs, they can determine where the data ends and markup resumes by accessing a table of significant delimiters provided by the parser, by using "length codes" embedded in the text, or a combination of the two.

2.11 Customizing

Many of the characteristics of the Standard Generalized Markup Language can be tailored to meet special needs.

2.11.1 The SGML Specification

There are two main categories of tailoring:

1. *Extensions* involve the use of optional features of the markup language.

   There are fourteen optional features that can be used:

ATTEXT	Attribute extension
ATTMIN	Attribute minimization
CENDTAG	Contained end-tag minimization
COSTRUC	Coexistent structure
DATATAG	Data tag minimization
DELIM	Delimiter modification
ENTITY	Dynamic entity definition
ID	Unique identifier attribute
NULLTAG	Null tag minimization
OMITTAG	Omitted tag minimization
RANK	Explicit rank
RECORD	Record boundary recognition
SDENTITY	Self-delimiting entity reference
STRUC	Dynamic structure and attribute definition

   They are described in "3 Extended Concepts" on page 34.

2. *Variations* are changes to the normal characteristics of a document as defined in the Standard.

   Variations include:

   a. Changing the quantitative characteristics, such as the maximum length of names and attribute values.

   b. Changing the name characters and declaration keywords to meet national language requirements.

   c. Changing the initial delimiter assignments.

   d. Using BITDATA or UCSDATA representations.

The tailoring of a document is described in its *SGML specification*, a statement (like a markup declaration) that appears in the *prolog* of a document, prior to the first start-tag. The **SGML** specification is normally provided automatically by the markup parser, but if a document's tailoring differs from what a processing system expects, the **SGML** specification must be entered explicitly.

Further details can be found in "4.11 SGML Specification" on page 74.

2.11.2 System-issued Declarations

A system can issue markup declarations in the prolog automatically, to create a tailored environment for the documents it processes. It could, for example, issue declarations for commonly used entity references or to change delimiter assignments. When a document is sent to a system with different characteristics, such declarations must be included explicitly.

2.12 Conformance

The Standard recognizes three categories of conforming document:

1. A document that has no variations and uses no features is termed a "basic SGML document." Its **SGML** specification will be like the one illustrated in "Basic SGML Document" on page 78.

2. A document that has no variations but does use an optional feature is called an "extended SGML document."

3. An SGML document, basic or extended, that has variations permitted by the Standard is termed a "variant" SGML document.

A basic SGML document will obviously be interchangeable more easily than one with extensions or variations, but will not be suitable for direct keying because it allows no minimization. In practice, most directly keyed documents will incorporate one or more of the optional features.

The Standard also recognizes categories of conforming systems, based on the characteristics of the conforming documents they can process. As the manner of implementing a processor of SGML documents is not specified by the Standard, there are no "conforming implementations."

3 Extended Concepts

A basic SGML document has the following *feature use* parameters on its **SGML**specification:

```
FEATURE  ATTEXT      NO
         ATTMIN      NO
         CENDTAG     NO
         COSTRUC     NO
         DATATAG     NO
         DELIM       FIXED
         ENTITY      PROLOG
         ID          NO
         NULLTAG     NO
         OMITTAG     NO
         RANK        NO
         RECORD      NO
         SDENTITY    NO
         STRUC       PROLOG
```

Optional features of the markup language are enabled by changing these parameters, as described in this clause.

3.1 Markup Environments

When the production of a document is planned in advance and carried out by a single person or a group working together, there should be no conflict in the use of entity names, unique identifiers, or other markup constructs. When several independently produced documents must be combined into a larger one, however, as in the publication of an anthology, the possibility of conflict does arise.

To avoid such conflict, markup environments can be established in marked sections of a document. Within a markup environment, delimiter assignments, entity names, unique identifiers, and structure and attribute declarations can be established independently of the document outside the environment.

A markup environment is established by a "DELIM", "ENTITY", "ID", "RANK", or "STRUC" parameter on a marked section start, which is in turn enabled only if "ENVIRON" was specified for the feature of the same name on the **SGML** specification. The details are described below in conjunction with descriptions of the individual features.

3.2 ENTITY: Dynamic Entity Definition

An **ENTITY** declaration is normally allowed only in the document prolog (that is, before the first start-tag). If "ENTITY CONTENT" is specified on the **SGML** specification, **ENTITY** declarations can also occur in the content of elements.

A specification of "ENTITY ENVIRON" implies "ENTITY CONTENT" and also allows *entity environments* to occur. At the start of an entity environment, the effect of any **ENTITY** declaration issued since the end of the prolog is canceled; it is restored at the end of the environment. The effect of any **ENTITY** declaration issued within the environment is canceled at the environment's end.

3.3 STRUC: Dynamic Structure and Attribute Definition

This feature is the same as the "ENTITY" feature, except that **STRUC** and the long form of **ATT** are the affected declarations.

3.4 DELIM: Delimiter Modification

The Standard Generalized Markup Language does not compel the use of particular characters (i.e., bit combinations) as delimiters. Instead, it defines delimiter *roles* and establishes initial character string assignments for them.

The assignments are normally fixed, but if "DELIM PROLOG" is specified they can be changed in the prolog, and if "DELIM CONTENT" is specified they can be changed in the content of elements in the document. The assignments can be changed to suit a particular keyboard or display, personal preferences, or the conventions of a particular text processing system.

3.4.1 Changing Markup Delimiters

Delimiter assignments are changed with a delimiter declaration (**DELIM**):

```
<!DELIM REP esmo "[" esmc "]" >
```

The "REP" parameter means the specified string is to replace the existing delimiter string; “esmo” and “esmc” are delimiter roles and the literals are the character strings to be assigned to the roles. (A string must normally be the same length as the string assigned to the role by the Standard, unless the length was changed by an **SGML** specification variation.)

The example replaces the assignments for the ***esmo*** and ***esmc*** delimiters with square brackets, causing descriptive tags to look like this:

```
[chapter id=chap1][section][p]The text begins ...
```

A delimiter role can be canceled altogether by specifying the “delete” ("DEL") parameter instead of “replace” ("REP"):

```
<!DELIM DEL esmo esmc >
```

If "REP" is specified without a character string, the parser will use the original string that existed at the end of the *prolog* (or at the start of the current *delimiter environment*, if any):

```
<!DELIM REP esmo esmc >
```

3.4.2 Concurrent Delimiter Assignments

Normally, only one string can be assigned to a delimiter role at a time, but the **SGML** specification delimiter use parameter could allow more:

```
DELIM    CUSTOM
--       ROLE     STRING  LEN  CNT  --
         pio      "<?"    2    2
         pic      ">"     2    2
         vi       "="     1    2
```

The specification allows two strings to be assigned concurrently to the ***pio***, ***pic***, and ***vi*** delimiter roles.

The following example lets a processing instruction begin or end with two question marks, in addition to the strings defined in the **SGML** specification:

```
<!DELIM ADD pio "??" pic "??" >
```

Given that declaration, the second through fifth records in the following example will be recognized as processing instructions:

```
Text preceding first processing instruction.
??sk 1??
<?in 3>
??in 5>
<?sk 2??
Text following fourth processing instruction.
```

A single one of several concurrent assignments can be deleted by specifying the string:

```
<!DELIM DEL pio "??" pic "??" >
```

The above declaration cancels the additional assignment, so that only the third record in the previous example will be recognized as a processing instruction.

When concurrent delimiters are allowed, a group of literals (connected by the ***or*** connector) can be specified in place of a single literal:

```
<!DELIM REP vi (":" | "=" ) >
```

The above example allows either the colon or the equal sign to be used as the ***vi*** delimiter. It is the equivalent of entering

```
<!DELIM REP vi ":" >
<!DELIM ADD vi "=" >
```

3.4.3 Selection Criteria

The criteria for selecting strings to be used as delimiters are fairly complex. They are described in detail in "Roles and Assigned Strings" on page 70. The basic principles, though, can be summarized briefly as follows:

1. Delimiter roles are classified by mode of use: for example, delimiters that occur in content, in an attribute value, in a declaration parameter, and so on. The character string assigned to a delimiter must be different from the strings assigned to the other delimiters in that class.

2. A delimiter string cannot contain a name.

3.4.4 Delimiter Environments

When "DELIM ENVIRON" is specified, a *delimiter environment* can be established by a marked section. The effect of any **DELIM** declaration issued within the environment is canceled at the end.

3.5 RECORD: Record Boundary Recognition

This feature allows record boundaries to be significant in document markup. When "RECORD YES" is specified, the parser declares the following entities to allow references to record boundary conditions in character or numeric literals:

&#RS. Record Start.
&#RE. Record End.
&#NR. Null Record.

When the delimiter modification feature is used, the following declaration lets a processing instruction begin with a period at the start of an input record and end at the end of the record:

```
<!DELIM REP pio "&#RS.." pic "&#RE." >
```

Note the two periods after "&#RS": the first is the entity reference close delimiter; the second is a data period. Given that declaration, the second and third records in the following example will be recognized as processing instructions:

```
Text preceding first processing instruction.
.sk 1
.in 3
Text following second processing instruction.
```

3.6 SDENTITY: Self-delimiting Entity Reference

The self-delimiting entity feature allows entity references to be undelimited character strings rather than delimited names. The feature is intended for use in conjunction with the record boundary recognition feature to support word processing entry conventions.

A self-delimiting entity is declared by specifying a literal instead of an entity name. The following declaration defines a null record as a substitute paragraph start-tag:

```
<!ENTITY "&#NR." "<p>" >
```

Or two horizontal tabs at the start of a record could stand for a paragraph start-tag:

```
<!ENTITY H/&#RS./09/09 "<p>" >
```

The two previous examples could be combined into one:

```
<!ENTITY ("&#NR." | H/&#RS./09/09) "<p>" >
```

A self-delimiting entity reference is declared and used like a delimited reference, except that it cannot be used in attribute values or parameters.

The maximum length and number of self-delimiting entity references is specified by the **SGML** specification when the feature is enabled. For example, the parameter

```
SDENTITY YES 4 8
```

allows a self-delimiting entity to be up to 4 characters long, and up to 8 self-delimiting entities to exist concurrently.

3.7 Structural Markup Minimization

A type definition establishes the possible variations in structure among documents of its type. This information can let the end user key less markup than he otherwise would have to.

3.7.1 The Concept of Minimization

Assume that a class of magazine article documents, with a GI of "article," has the following type definition:

```
<!STRUC --
ELEMENT   CONTENT --
article   (title, body)
title     (&#CHAR.*)
body      (p*)
p         (&#CHAR. | list)*
list      (item*)
item      (p | list)*
>
```

The full markup for an individual article might look something like this:

```
<article>
<title>The Cat</title>
<body>
<p>A cat can:
<list>
<item>jump</item>
<item>memo</item>
</list>
</p>
<p>It has 9 lives.
</p>
</body>
</article>
```

Note that the type definition says that a paragraph cannot immediately contain another paragraph; it can contain only text characters and lists. Similarly, a list item cannot immediately contain another list item (even though it could contain another list). As a result, the markup can be minimized by omitting many of the end-tags:

```
<article>
<title>The Cat</title>
<body>
<p>A cat can:
<list>
<item>jump
<item>meow
</list>
<p>It has 9 lives.
</article>
```

It is possible to omit the item end-tag: the occurrence of another item indicates the end of the previous one (because an item cannot contain an item). The paragraph end-tag is omittable by the same reasoning (a paragraph cannot contain another paragraph).

Finally, it is logical that the end of an element should also be the end of everything it contains. In this way, the article end-tag ends the body and the last paragraph as well as the article.

3.7.2 Specifying Minimization

Markup minimization is a good thing, but not if it makes it harder to detect markup errors. In the last example, had the list end-tag been omitted, it would have been implied by the article end-tag, just as the body end-tag was.

```
<article>
<title>The Cat</title>
<body>
<p>A cat can:
<list>
<item>jump
<item>meow
<p>It has 9 lives.
</article>
```

This would not have been the author's intention, though, as the last paragraph would erroneously have been made part of the last item in the list.

To prevent such misinterpretations, there are optional parameters on the **STRUC** declaration that specify the minimization that is allowed for an element. There are several types, any or all of which could be specified separately for the start-tag and end-tag. They are enabled by four optional features, which are selected by specifying "YES" on the **SGML** specification.

3.7.3 OMITTAG: Omitted Tag Minimization

An end-tag is omittable when it is followed by the start-tag of an element that cannot occur within it. This was the case for the paragraph and item end-tags in the article example. The **STRUC** declaration would look like this:

```
<!STRUC --
ELEMENT  MIN?  CONTENT  --
p        - O   (&#CHAR. | list)*
list           (item*)
item     - O   (p | list)*
>
```

and have the following meaning:

1. The "MIN?" heading identifies a pair of parameters for start-tag and end-tag minimization. Either both parameters must be specified or neither.

2. The "O" indicates that omitted tag minimization is allowed for the end-tag of the "p" and "item" elements.

3. The hyphen is the ***less*** delimiter; it indicates that no minimization is allowed for the start-tags. (Without it, the "O" would be taken for the start-tag minimization parameter.)

An end-tag can also be omitted if an element has a null content, because anything that follows the start-tag must be part of the containing element.

```
<!STRUC --
ELEMENT  MIN?  CONTENT  --
figref   - O   ()
>
```

A start-tag is omittable when only one possibility is allowed. For example, a list could be marked up like this:

```
<list>
This is the first item (what else ?)</item>
This is the second item.</item>
This is the third and last item.</item>
</list>
```

The first item is implied by the start of the list. Each of the others is implied by the end-tag of the item that precedes it.

The **STRUC** declaration would look like this:

```
<!STRUC --
ELEMENT  MIN?  CONTENT --
list           (item*)
item     O -   (&#CHAR.+)
>
```

When combined with the end-tag "O" minimization, the **STRUC** declaration would be as follows:

```
<!STRUC --
ELEMENT  MIN?  CONTENT --
list           (item*)
item     O O   (&#CHAR.+)
>
```

3.7.4 CENDTAG: Contained End-tag Minimization

A contained element's end-tag is omittable when it is followed by the end-tag of an element that contains it. "E" end-tag minimization must be specified for the element whose end-tag is omittable and "C" end-tag minimization for the containing element:

```
<!STRUC --
ELEMENT  MIN?  CONTENT --
list     - C   (item*)
item     - E   (&#CHAR.+)
>
```

The above declaration allows the end-tag of the third item to be omitted from the following list, because it is implied by the list end-tag:

```
<list>
<item>This is the first item (what else ?)</item>
<item>This is the second item.</item>
<item>This is the third and last item.
</list>
```

Combined with the "O" minimization discussed previously, the **STRUC** declaration would be:

```
<!STRUC --
ELEMENT  MIN?  CONTENT --
list     - C   (item*)
item     O EO  (&#CHAR.+)
>
```

and the list could be marked up like this:

```
<list>
This is the first item (what else ?)
<item>This is the second item.
<item>This is the third and last item.
</list>
```

3.7.5 NULLTAG: Null Tag Minimization

The null tag feature enables "N" minimization and some special delimiters for use with it.

A null tag is one in which no descriptive markup is specified. Instead, the parser assumes that the tag's descriptive markup is the same as that of the most recently started element. For example, the following would be equivalent with "N" minimization:

```
This is a <q>quoted</q> word.
This is a <q>quoted</> word.
```

And a list could become the following:

```
<!STRUC --
ELEMENT  MIN?  CONTENT --
list           (item*)
item     N N   (p | list)*
>
<list>
<item>This is the first item (what else ?)</>
<>This is the second item.</>
<>This is the third and last item.</>
</list>
```

"N" minimization can be combined with other kinds, as in the following example:

```
<!STRUC --
ELEMENT  MIN?    CONTENT --
p        -  O    (&#CHAR. | list)*
list     -  C    (item*)
item     NO ENO  (p | list)*
>
<list>
This is the first item (what else ?)
<>This is the second item.
<>This is the third and last item.
</list>
```

Note that when the first null tag was encountered, "item" and not "list" was the open element, even though "item" was implied by "O" minimization rather than entered explicitly. It was also necessary to say "</list>" instead of "</>" at the end, because the latter would have meant "item end-tag," rather than "list end-tag."

Had "N" end-tag minimization been specified for "list" as well, "</></>" could have been entered to mean "end the item, then end the list." The declaration would have been:

```
<!STRUC --
ELEMENT  MIN?    CONTENT --
p        -  O    (&#CHAR. | list)*
list     -  CN   (item*)
item     NO ENO  (p | list)*
>
```

Markup can be reduced further by using the single character delimiters that are enabled by the null tag feature. The delimiters are the "null start-tag" (***nst***), normally a percent sign, and the "null end-tag" (***net***), normally the greater-than symbol.

```
This is a <q>quote> with a net delimiter.
This is a list with nst delimiters
for the items:
<list>
This is the first item (what else ?)
%This is the second item.
%This is the third and last item.
</list>
```

The disadvantage is that these delimiter characters cannot be used directly as data; they will always be interpreted as delimiters unless they are disabled by a **DELIM** declaration.

```
<!DELIM DEL net nst >
```

It is possible, however, to declare unparsable entities for the characters:

```
<!DELIM DEL net nst -- Disable delimiters temporarily. -- >
<!ENTITY percent MS><!MS DATA>%<!MSE>
<!ENTITY gtsym   MS><!MS DATA>><!MSE>
<!DELIM REP net nst -- Enable delimiters again. -- >
```

The characters can then be entered indirectly as data by means of an entity reference:

```
This is a <q>quote with a greater-than
(&gtsym.) symbol> in it.
```

The need for an entity reference can largely be overcome for null end-tags by using a "conditional null end-tag" delimiter (***cnet***). The ***cnet*** is normally a vertical bar (|). It is interpreted as a null end-tag only for an element in which it was used as (i.e., in place of) the ***esmc*** delimiter.

```
<p>This paragraph has
a <q|quotation| in it and
a vertical bar (|) that is data.</p>
```

The following list summarizes the delimiters enabled by the null tag feature and the character strings that the Standard recommends for them:

String	Name	Meaning
|	CNET	Conditional null end-tag.
>	NET	Null end-tag.
%	NST	Null start-tag.

3.7.6 DATATAG: Data Tag Minimization

The data tag minimization feature allows literals in a content model to define the data characters that can occur. The feature also enables "D" minimization, which permits such data characters to be interpreted as tags while remaining part of the data content.

3.7.6.1 Literals in Models

Normally, an element's content model can contain only element GI's, references to them, and references to four parser-declared parameter entities: #CHAR, #CSDATA, #BITDATA, and

#UCSDATA.

For some elements it is desirable to be more specific about the data characters that can occur. If a textbook, for example, had exercise numbers like “2-15” and “12-2,” the exercise number (“exnum”) would normally be modeled in one of the following ways:

```
<!STRUC exnum (&#CHAR.+)   >
```

or

```
<!STRUC exnum (&#CSDATA.) >
```

The first declaration would allow entity references and other markup to occur in an exercise number while the second would not. Both of the models, though, would let any characters constitute an exercise number.

In contrast, the following model would require the exercise number to contain precisely two decimal numbers separated by a hyphen:

```
<!STRUC exnum (&#NUM., "-", &#NUM.) >
```

The “&#NUM.” is a reference to a parameter entity, declared by the parser, with the meaning:

```
("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9")+
```

that is, “a decimal number with 1 or more digits.” The quoted hyphen is a character literal.

The model with literals is more useful than the two earlier models because it supplies the markup parser with sufficient information to detect errors in the content. The “exnum” procedure can therefore be made simpler, because it can process the content without having to check syntactic validity.

The data tag feature can also be used in conjunction with the attribute extension feature to allow arbitrary literals in attribute value models (discussed later).

3.7.6.2 Data Tags

The techniques of markup minimization discussed so far go a long way toward creating text that appears almost free of markup. The following list, for example, has no visible explicit markup within it:

```
<!DELIM  REP net "&#RE." >
<!STRUC --
ELEMENT  MIN?  CONTENT --
joblist        (jobitem+)
jobitem  O N   (&#CHAR.+)
>
<joblist>
Sharon Adler,          Vice Chairman
Larry Beck,            Secretary
Aaron Bigman,          Member
Jim Cox,               Member
Bill Davis,            Chairman
Joe Gangemi,           Member
Charles Goldfarb,      Editor
Mike Goodfellow,       Consultant
Randy Groves,          Past Member
Charles Lightfoot,     Past Member
Sperling Martin,       Member
Bettie McDavid Mason, Consultant
Jim Mason,             Member
Norm Scharpf,          Observer
Ed Smura,              Member
Bill Tunnicliffe,      Member
Kit von Sück,          Member
</joblist>
```

The start-tag for each "jobitem" is omittable by "O" minimization. The end-tags are present, but they are null tags represented by the ***net*** delimiter. The **DELIM** declaration defines the ***net*** as a record end, which is why the end-tag markup, though explicit, is not visible.

The **STRUC** declaration states that a "jobitem" is merely a sequence of characters. A reader, however, sees it as containing separate "name" and "job" elements because the comma and spaces identify the job title as clearly as a start-tag would have. Such a situation, in which the conventions used in the textual data can unambiguously identify elements within it, is known as *data tag markup*.

The Standard Generalized Markup Language data tag feature allows these conventions to be expressed in a **STRUC** declaration through the use of literals:

```
<!STRUC --
ELEMENT  MIN?  CONTENT --
jobitem  - C   (name, (",", " "+), job)
name     O O   (&#CHAR.+)
job      D E   (&#CHAR.+)
>
```

The declaration states that:

1. The end-tag for a "jobitem" will terminate a "job" ("C" end-tag minimization).
2. The start-tag for a "name" is omittable because it must be the first thing in a "jobitem" ("O" minimization).
3. The "name" end-tag is omittable because a "job" element is not allowed in a "name" ("O" minimization).
4. The "job" start-tag is omittable because it is implied by the comma and spaces in the "jobitem" ("D" minimization).
5. The "job" end-tag is omittable because it is implied by the end of the "jobitem" ("C" minimization).

The complete declaration for the "joblist" would appear as follows:

```
<!DELIM  REP net "&#RE." >
<!STRUC --
ELEMENT  MIN?  CONTENT --
joblist        (jobitem+)
jobitem  O CN  (name, (",", " "+), job)
name     O O   (&#CHAR.+)
job      D E   (&#CHAR.+)
>
```

Characters are normally either data or markup, but not both. The essence of "D" minimization is that characters are both data and a tag at the same time, i.e., a *data tag*. A data start-tag must conform to a nested *literal model* that precedes the data tag's element in its containing element's model. In the above example, the literal model is

```
(",", " "+)
```

which means "a comma followed by one or more blanks." Thus, any string consisting of a comma and one or more blanks that follows a "name" in a "jobitem" will be construed as a "job" start-tag. It will also, however, be considered part of the data of the "jobitem."

Note the differences between a data tag and a self-delimiting entity reference whose entity is a tag:

1. The entity reference string is markup, not data.

2. The entity reference is recognized anywhere; the data tag is recognized only where a **STRUC** declaration allows it.

3. The entity reference is a constant string; the data tag can be any of a number of strings that conform to the literal model.

The above example could also have been set up to declare the string to be a data end-tag for the "name,"

```
<!STRUC --
ELEMENT  MIN?  CONTENT --
jobitem  O CN  (name, (",", " "+), job)
name     O D   (&#CHAR.+)
job      O E   (&#CHAR.+)
>
```

or to define separate data tags for both elements:

```
<!STRUC --
ELEMENT  MIN?  CONTENT --
jobitem  O CN  (name, (","), (" "+), job)
name     O D   (&#CHAR.+)
job      D E   (&#CHAR.+)
>
```

Data tag minimization is useful for applications that analyze text. The following example uses data tags to identify sentences and words within a paragraph:

```
<!ENTITY #stop "(D/46|D/63|D/33|D/58)" --("."|"?"|"!"|":")-->
<!STRUC --
ELEMENT   MIN?  CONTENT --
p               (sentence, (&#RE.))*
sentence O D    ((word, (&#S.+))*, word, (&#stop.))+
word     O D    (&#CHAR.+)
>
<p>The first sentence ends here.
The second sentence ends
here.
This isn't a whole sentence.  The
third sentence ends "not here!," but here!
</p>
```

In the example, a word end-tag is either one or more separator characters, or, if it is the last word in a sentence, a "stop" character. The sentence data end-tag is recognized when the last word in the sentence (i.e., the one terminated by a stop character) is followed by a record end.

3.7.7 Structural Markup Minimization Considerations

"E" and "N" minimization require the user (and the markup parser) to know the current location in the document's structure. "D" and "O" minimization, though, also require knowledge of the **STRUC** declarations.

This difference should be considered in deciding what kind of minimization the user can handle without introducing errors. Moreover, if the implementation makes the procedures handle the type definition processing, minimization might make the procedures too complex.

3.8 RANK: Explicit Rank

The rank of an element is its level of nesting. In many document types, rank is only implied by the beginning and ending of nested elements, such as lists, and is never specified explicitly in the document markup.

Some markup designers, though, prefer to use explicit rank designations for some elements, such as headings and paragraphs (e.g., p1, p2). Such elements tend to have structure declarations like:

```
<!STRUC --
ELEMENT   MIN?  CONTENT --
p1        - O   (&#CHAR.+, p2*)
p2        - O   (&#CHAR.+, p3*)
p3        - O   (&#CHAR.+, p4*)
p4        - O   (&#CHAR.+)
>
```

Not only is the example repetitious, but it requires a limit on the highest rank number (here, 4). (The rank numbers, incidentally, run inversely to the rank levels: p1 is at the highest level, and p4 the lowest.)

Specifying "RANK YES" enables the explicit rank feature, which offers a more convenient way of specifying rank explicitly. An element can be designated a *ranked element* by attaching a *rank number* to its GI in a structure declaration:

```
<!STRUC --
ELEMENT   MIN?  CONTENT --
p[]       - O   (&#CHAR.+, p[+1]*)
>
```

The square brackets are the "rank open" and "rank close" delimiters (***ranko*** and ***rankc***). Within the brackets, in each case, is the rank number, which can be absolute, relative, or null. A null rank number means "any rank." So this declaration means that a paragraph of any rank contains characters and can also contain any number of paragraphs of the next highest rank number (i.e., the next level down).

A ranked element is marked up in a document by appending the rank number to the GI (but without the brackets). If a rank number is not entered, the element's last specified rank number is used.

```
<p1>Text of 1st level paragraph.
<p>Another 1st level paragraph.
<p2>Nested 2nd level paragraph.
<p>Another 2nd level paragraph.
<p1>Back to 1st level.
```

Within a rank expression in a **STRUC** declaration, an ellipsis ("...") can be used as a ***range*** delimiter. It indicates that the ranks referred to range from the number before the delimiter to the rank number after it. As before, a null rank number means "any rank" in that direction.

The example below allows a p1 to contain only text and p2s, but a p2 through p4 can contain any lower level paragraphs, through level 5.

```
<!STRUC --
ELEMENT    MIN?  CONTENT --
p[1]       - O   (&#CHAR.+, p[2]*)
p[2...4]   - O   (&#CHAR.+, p[+1...5]*)
>
```

A group of element types can share the same rank if they have identical content models. Such a group is called a *ranked group*. For example, if a document had normal paragraphs, numbered paragraphs, and bulleted paragraphs, each might contain any of the three at the next level down. The declaration

```
<!STRUC --
ELEMENT         MIN?  CONTENT --
(p | n | b)[] - O    (&#CHAR.+,(p[+1] | n[+1] | b[+1])* )
>
```

lets a document contain:

```
<p1>Text of 1st level paragraph.
<n>Numbered 1st level paragraph.
<p2>Nested 2nd level paragraph.
<b>Bulleted 2nd level paragraph.
<p1>Back to 1st level.
```

Relative rank numbers (both positive and negative) can also be used with GIs. If the sign alone is used, it means plus (or minus) one. The previous example could therefore have been marked like this:

```
<p1>Text of 1st level paragraph.
<n>Numbered 1st level paragraph.
<p+1>Nested 2nd level paragraph.
<b>Bulleted 2nd level paragraph.
<p->Back to 1st level.
```

If "RANK ENVIRON" is specified, the explicit rank feature is enabled and it also becomes possible to specify a "RANK" environment on a marked section. Explicit rank numbers used in the environment are treated as relative to the ranks that existed when the environment began.

The following list summarizes the delimiters enabled by the explicit rank feature and the character strings that the Standard recommends for them:

Char	**Name**	**Meaning**
[	**RANKO**	Rank expression, open.
]	**RANKC**	Rank expression, close.
...	**RANGE**	Indicates a range of values.
+	**PLUS**	Increment rank number (decrement nesting level).
-	**MINUS**	Decrement rank number (increment nesting level).

3.9 ATTMIN: Attribute Minimization

Some attribute values imply their names. The example of a memo in Figure 2 on page 17 had a "status" attribute whose value was either "draft" or "final." The markup was unnecessarily cumbersome, since "draft" implies "status."

The attribute markup minimization feature allows the attribute name and ***vi*** to be omitted in such cases if a "minimization" parameter is specified on the attribute declaration:

```
<!ATT memo --
NAME   MIN? VALUE              DEFAULT --
status A    ("FINAL"|"DRAFT")  "FINAL"
>
<memo draft>
```

Minimization can be specified only when the model is a group of literals separated by ***or*** connectors. The literals must be *name tokens*; that is, they must be composed of name characters and cannot exceed the allowed length of a name. None of the name tokens can appear in the model of any other minimizable attribute in the same declaration, or as the name of an attribute.

Values of minimizable attributes are translated to uppercase in the same way as names, regardless of whether they are entered with the attribute name and ***vi***. As a reminder of the translation, it is good practice to enter the name tokens in uppercase in the ATT declaration.

Minimization inherently makes the document markup more ambiguous. This effect can be ameliorated if the groups are kept small and the name tokens chosen are descriptive adjectives that imply the attribute name (e.g., "NEW | REVISED", "CONFIDENTIAL | INTERNAL | GENERAL"). In the following example, the "compact" attribute is fine from this standpoint, but "emphasis" is only marginally acceptable:

```
<!ATT list --
NAME       MIN?  VALUE                DEFAULT --
compact    A     ("COMPACT")          UNKNOWN >
<!ATT phrase
emphasis   A     ("0"|"1"|"2"|"3")    "0"      >
```

3.10 ATTEXT: Attribute Extension

The attribute extension feature allows complex combinations of text, elements, and unparsable data to occur in attribute values.

3.10.1 Complex Value Models

An attribute value must normally conform to one of the following forms of model:

```
(&#CSDATA.)
(&#CHAR.*) or (&#CHAR.?) or (&#CHAR.+)
("nametoken" | "nametoken"
 | etc. )
(&#NAME.)
(&#NAMES.)
(&#NUM.)
(&#NUMS.)
```

If "ATTEXT YES" is specified, however, the model can have nested groups with arbitrary connectors and contain element names. For example, in “Complex Attribute Values” on page 19, attributes were discussed whose values logically consist of a number of elements. The example used was a computer output display with two associated colors, green and black.

With the extended attribute feature, the “color” attribute could be defined to be a structure:

```
<!ATT    display  color        (bg, fg)        UNKNOWN >
<!STRUC           (bg|fg)  E   (&#NAME.)           >
<!>
<display color="<bg>black<fg>green">
```

If "DATATAG YES" were also specified, an attribute value model could include literals. If "BITDATA YES" or "UCSDATA YES" were specified, references to #BITDATA and #UCSDATA, respectively, could also be used.

3.10.2 Redundant AVC Delimiters

When structures can occur in attribute values, a user error in omitting the ***avc*** could prevent the remainder of the document from being processed properly. For this reason, the attribute extension feature allows the ***avc*** to be used redundantly as an attribute separator to permit the markup parser to recover if a required ***avc*** is inadvertently omitted.

In the following example of a start-tag for a figure, the caption requires delimiters. If the ***avc*** were omitted, the remainder of the source file would be treated as part of the caption. However, the redundant ***avc*** character that precedes the “place” attribute would keep this from happening by terminating any open attribute value. An ***avc*** is also shown preceding “frame” because, although it would not be helpful in this instance, a user might want to be in the habit of preceding every attribute name with an ***avc*** to insure safety.

```
<figure
caption="An attribute value can be a complex
structure, with:
<list>
<item>Text characters.
<item>Unparsable data.
<item>Nested elements.
</list>"
" place=float
" frame=none
>
```

The ability to use redundant ***avc*** delimiters is intended to make markup clear and error-free, but it can be abused:

```
<memo "sender=jones" version="02" "required>
```

The double quotation marks in the above example have the following meanings:

1. Redundant ***avc*** preceding "sender" attribute.
2. Redundant ***avc*** preceding "version" attribute.
3. Required ***avo*** for "version" attribute value.
4. Required ***avc*** for "version" attribute value.
5. Redundant ***avc*** preceding value of minimized "status" attribute.

3.11 ID: Unique Identifier Attribute

A *unique identifier* ("ID") is an attribute that names an element to distinguish it from all other elements. Specifying "ID YES" enables the ID feature, which lets the markup parser perform common processing for ID attributes, thereby minimizing the implementation effort for procedures.

The purpose of IDs is to allow one element to refer to another: for example, to allow a figure reference to refer to a figure. Normally, the procedure for the figure would associate some data with the ID (such as the figure number). The figure reference procedure would retrieve the data and print it.

Although the markup parser is normally unaware of the meaning of particular attributes, it can be told when an attribute is a unique identifier:

```
<!ATT memo --
NAME        MIN?  VALUE                  DEFAULT --
id                (&#ID.)                UNKNOWN >
```

Only one ID attribute is allowed for an element.

The value of a unique identifier attribute must be a name that is different from the value of any other ID attribute in the same *ID environment*. (The ID environment is normally the document, but subordinate environments can be established in a manner that will be discussed later.) As with other names, the uppercase form is used, regardless of how it was entered.

The element that does the referencing must have a "unique identifier reference" attribute, indicated in the **ATT** declaration model by the entity reference "&#IDREF.":

```
<!ATT figref --
NAME        MIN?  VALUE                  DEFAULT --
refid             (&#IDREF.)             UNKNOWN >
```

The value of an ID reference attribute must be a name that is the same as that of an ID attribute in the same ID environment or in an ID environment that is subordinate to it. As usual, the uppercase form of the name is employed.

ID environments are enabled by specifying "ID ENVIRON" on the **SGML** specification, which also enables the ID feature. The specification allows an ID environment to be created for the duration of a marked section:

```
<!MS (ID)>...content of marked section...<!MSE>
```

Unique identifiers defined within the section are maintained independently of those outside of it, and can therefore duplicate them. Resolution of ID references within the section is attempted first with IDs defined in the section, then with IDs defined in ID environment marked sections nested within the section, in the order the nested sections occur in the document.

3.12 COSTRUC: Coexistent Structure

The discussion thus far has been confined to markup for the logical, or abstract, structure of a source document that is to be processed. However, the result of processing a document is also a document, albeit one that might have a radically different structure.

For example, a one-dimensional source document that is composed of chapters, sections, and paragraphs, will, after formatting, become a two-dimensional result document whose elements are pages, columns, and lines. The two structures will coincide at certain points, certainly at the highest (document) level and possibly at others. A chapter in the source, for instance, might correspond to a "page group" in the result.

It is sometimes useful to maintain information about a source and a result document simultaneously in the same entity, as in "what you see is what you get" (WYSIWYG) word processors where the user interacts with the formatted output but the editorial changes are reflected back to the source as well.

There are also cases in which even more than two "views" of the document can be useful, such as maintaining multiple formatted results for instant display on both a CRT and a printer while still having the logical document available for other applications.

The coexistent structure feature supports multiple coexistent structural views in addition to the abstract view by allowing the user to define special ***esmo*** and ***eemo*** delimiters. They are known as the "coexistent structure, element start" (***cnes***) and "coexistent structure, element end" (***cnee***) delimiters. In each case, the *n* is replaced by a decimal number, so the delimiter roles are actually called ***c1es***, ***c2es***, and so on.

With the coexistent structure delimiters, multiple alternate sets of start-tags and end-tags can be used in a document in addition to the base set. Other delimiters are the same for coexistent structures as they are for the base structure.

The structure and attribute declarations allow the user to specify whether they refer to one of the coexistent structures or to the base structure.

4 Formal Specification

4.1 Notation and Conventions

The notation for syntax productions is as follows:

1. Occurrence of syntactic tokens:

   ? Optional (0 or 1 time).
   + Required and repeatable (1 or more times).
   * Optional and repeatable (0 or more times).

2. Ordering of syntactic tokens:

   , One of each, in the order entered.
   @ One of each, in any order.
   | One only.

3. Other notation:

   = Is defined as:
   ( ) Used for grouping.

   The occurrence indicators are applied first, then the ordering connectors. Parentheses can be used as in mathematics to change these priorities.

Within syntax productions, and when syntactic constructs are referenced in the text, the following formatting conventions are used:

syntactic variable
"syntactic literal"
delimiter role
TERMINAL SYMBOL

A "delimiter role" is a delimiting function that is performed in a document by a character string assigned to the role. An alphabetical list of delimiter roles, their functions, and the strings normally assigned to them is provided in "Roles and Assigned Strings" on page 70.

A "terminal symbol" refers to one or more bit combinations in the code of the document's character set. Terminal symbols are defined as they are introduced in the formal specification.

4.2 Document

[1] document = *prolog*, *element*

[2] prolog = *SGML specification*, (*markup declaration* | *processing instruction* | *marked section* | *entity reference* | *parameter entity reference* | *s*)

Notes:

1. The *prolog* must include a *structure declaration* for the *element*.

2. The *prolog* cannot include a *delimiter declaration* if "DELIM FIXED" is specified on the *SGML specification.*

3. If a *marked section* occurs in the *prolog* and it is not being stored as an entity:

   a. The *marked section start* cannot specify "DATA".

   b. Everything in the *content* must be valid in a *prolog* outside of a *marked section.*

4. An *entity reference* or *parameter entity reference* is replaced by its entity. The result must be a syntactically valid *prolog.*

5. If a document is processed exclusively by a single system, the system can issue all or part of the *prolog* with the same effect as if it had been included explicitly in the document source. The entire *prolog*, however, must be included explicitly if the document is subsequently sent to another system.

4.3 Element

[3] element = *start-tag*?, *content*, *end-tag*?

Notes:

1. The *start-tag* can be omitted only when the element's *structure declaration* specifies *start-tag minimization* as:

   a. "O", and the element is the only type that could occur.

   b. "D", and a conforming data tag is present.

2. The *end-tag* can be omitted only when the element's *structure declaration* specifies *end-tag minimization* as:

   a. "O", and the element is followed by an element or character that is not allowed in its *content.*

   b. "D", and a conforming data tag is present.

   c. "E", and the element is followed by the *end-tag* of another element that contains it and whose *structure declaration* specifies "C "end-tag minimization.

   d. "D", "E", or "O", and the last character of the element is the last character of the *document* or *attribute value* in which the element occurs.

4.3.1 Start-tag

[4] start-tag = (***esmo***, *descriptive markup*?, (***esmc***? | ***cnet***)) | ***nst***

Notes:

1. The *descriptive markup* can be omitted, or the ***cnet*** or ***nst*** used, only when the element's *structure declaration* specifies "N" for *start-tag minimization*. In such cases, the most recently started but unended element's *descriptive markup*, exclusive of any unique identifier attribute, is used as the descriptive markup of this element.

2. The ***esmc*** can be omitted when *descriptive markup* is present and is followed immediately by another *start-tag*.

3. If the element's *structure declaration* specified a *coexistent structure*:
   a. A ***cnes*** delimiter will be used in place of ***esmo***.
   b. The ***nst*** cannot be used.

4.3.2 Content

[5] content = ***CSDATA*** | (***UCSDATA*** | ***BITDATA*** | *character* | *element* | *other markup*)*

[6] other markup = *entity reference* | *self-delimiting entity reference* | *parameter entity reference* | *markup declaration* | *marked section* | *processing instruction*

where:

CSDATA ("character set data") is data in the document's character set other than:

1. a ***SYSCHAR*** character; or
2. a character string assigned to a delimiter role that would delimit the end of the syntactic construct in which the ***CSDATA*** occurs; i.e.,
   a. In *content*, the ***net***, the ***cnet***, and the sequence consisting of the ***eemo*** followed by a ***NAMESTRT*** character.
   b. In an *attribute value*, the ***avc***.
   c. In a *ps* (parameter separator comment), the ***psc***.
   d. In a *character literal* that is being stored as an entity, the ***clc***.

Other delimiter roles within ***CSDATA*** are disabled. ***CSDATA*** can be null.

BITDATA ("bit combination data") is data composed of bit combinations that do not represent characters. It is not specified by this Standard. Users of "BITDATA" must define the means of determining when it starts and ends.

UCSDATA ("undefined character set data") is character data that is not in the document's character set. It is not specified by this Standard. Users of "UCSDATA" must define the means of determining when it starts and ends.

Notes:

1. An element's *structure declaration* determines whether ***CSDATA***, ***BITDATA***, ***UCSDATA***, *character*, and *element* are allowed in the *content* structurally, which affects (but does not of itself fully specify) what can occur in the *content* syntactically:

   a. The *other markup* can occur whenever ***CSDATA*** is not allowed structurally.

   b. If neither ***CSDATA*** nor *character* is allowed structurally, a *character* can occur only as part of an *element* or *other markup*, except that an extraneous ***SPACE***, ***RS***, ***RE***, or ***NR*** can occur, but is ignored.

   c. If *character* is allowed structurally, a *character* that is not part of an *element* (if allowed) or *other markup* is data. A data *character* can also be part of a data tag for an *element* with "D" minimization.

   d. If ***CSDATA*** is allowed structurally, a *character* can occur only as data.

2. If a data *character* is part of the sequence "***RE***, ***RS***" it will be ignored if the sequence immediately follows the element's *start-tag*, a *processing instruction*, a *marked section start*, or a *markup declaration*, or if it immediately precedes the element's *end-tag* or a *marked section end*.

3. If an element's *structure declaration* defines the *content* to contain one or more *names*, the uppercase form of each declared *name* is used, regardless of how it was entered.

4. An *entity reference*, *parameter entity reference*, or *self-delimiting entity reference* is replaced in the *content* by its entity. The result must be syntactically valid and must comply with the element's *structure declaration* for allowable *content*.

5. The kind of *markup declaration* that can occur depends on the *SGML specification*, as follows:

   a. A *delimiter declaration* is allowed only if "DELIM CONTENT" or "DELIM ENVIRON" is specified.

   b. An *entity declaration* is allowed only if "ENTITY CONTENT" or "ENTITY ENVIRON" is specified.

   c. A *structure declaration* or an *attribute declaration* with a *declared value* parameter is allowed only if "STRUC CONTENT" or "STRUC ENVIRON" is specified. An *attribute declaration* with no *declared value* parameter is allowed whenever *other markup* is allowed.

4.3.2.1 Character

[7] character set = ***SYSCHAR*** | *character*

[8] character = ***NAMECHAR*** | ***SPACE*** | ***PENTCHAR*** | ***BITCOM*** | *pseudocharacter*

[9] pseudocharacter = ***RS*** | ***RE*** | ***NR***

where:

SYSCHAR	is a bit combination reserved for system use.
NAMECHAR	is a bit combination from the subset of the document's *character set* that is allowed in a *name*.
SPACE	is a bit combination that represents a space.
PENTCHAR	is a bit combination from the subset of the document's *character set* that is allowed to begin a *parameter entity name*.

BITCOM	is any other bit combination.
RS	indicates the start of a source record.
RE	indicates the end of a source record.
NR	indicates a null source record with no ***RS***, ***RE***, or other *character*.

Notes:

1. Bit combinations are assigned to ***SYSCHAR***, ***SPACE***, ***PENTCHAR***, and ***NAMECHAR*** by by the *character use* parameter of the *SGML specification*. The remaining bit combinations in the document's code are assigned to ***BITCOM***.

2. The representation of a *pseudocharacter* is not specified by this Standard. The system must define how a *pseudocharacter* can be distinguished from any other *character*.

4.3.3 End-tag

[10] end-tag = (***eemo***, *descriptive markup*?, ***eemc***?) | ***net*** | ***cnet***

Notes:

1. The *descriptive markup* can be omitted, or the ***net*** or ***cnet*** used, only when the element's *structure declaration* specifies "N" end-tag minimization. There is a further restriction in that the ***cnet*** can be used only if the identical ***cnet*** delimiter string was used in the element's *start-tag*. In all of these cases, the most recently started but unended element's *descriptive markup*, exclusive of any unique identifier attribute, is used as the *descriptive markup* of this element.

2. The ***eemc*** can be omitted when *descriptive markup* is present and is followed immediately by a *start-tag* or another *end-tag*.

3. If the element's *structure declaration* specified a *coexistent structure*:

   a. A ***cnee*** delimiter will be used in place of ***eemo***.

   b. A ***net*** or ***cnet*** cannot be used.

4.4 Descriptive Markup

[11] descriptive markup = *generic identifier*, ((*s* | ***avc***)+, *attribute specification*)*, (*s* | ***avc***)*

Notes:

1. If the element's *structure declaration* specifies "N" start-tag or end-tag minimization, the length of *descriptive markup* cannot exceed the "PARMLEN" value on the *SGML specification*; if not, the length cannot exceed the "DESCLEN" value.

2. An ***avc*** can occur only if "ATTEXT YES" is specified on the *SGML specification*.

3. The *generic identifier* and each *attribute specification* in an end-tag's *descriptive markup* must have occurred in the same element's start-tag *descriptive markup*.

4. A *generic identifier* is valid:

   a. For an element occurring in *content*, if it was named in the *content model* of the *structure declaration* for the element in which it occurred.

   b. For an element occurring in an *attribute value*, if it was named in the *value model* of the *attribute declaration* for the element in which it occurred.

5. The validity of the *attribute specification* list is determined by the element's *attribute declaration*:

   a. There must be an *attribute specification* for each attribute declared as "REQUIRED".

   b. Only one *attribute specification* is permitted for a single attribute.

   c. If there is no *attribute specification* for an attribute that has a declared default value, an *attribute specification* with the default value is implied.

4.4.1 Generic Identifier (GI)

[12] generic identifier = *name*, *rank number*?

Notes:

1. A *rank number* can occur only if the element's *structure declaration* defines it to be a *ranked element* or a member of a *ranked group*:

   a. If *rank number* is null for such an element, the *rank number* of the most recent element of the same type or in the same *ranked group* is implied.

   b. A *character* that could be in either a *name* or a *rank number* is treated as though in the *rank number*.

2. The length of the *generic identifier* must not exceed the "NAMELEN" value specified on the *SGML specification*.

4.4.2 Attribute Specification

[13] attribute specification = (*name*, *s**, ***vi***, *s**)?, ***avo***?, *attribute value*, ***avc***?

Notes:

1. The *name* and ***vi*** can be omitted only if the element's *attribute declaration* specifies "A" minimization for that attribute. If they are omitted, the ***avo*** and ***avc*** must also be omitted.

2. If the element's *attribute declaration* specifies "A" minimization for an attribute, the uppercase form of the *attribute value* is used, regardless of whether the *name* and ***vi*** were specified.

3. If an *attribute value* has been defined by an *attribute declaration* as an *id value*, its attribute is termed a "unique identifier attribute" and:

   a. The *attribute value* must be a *name*.

   b. The uppercase form of the *attribute value* is used, regardless of how it was entered.

   c. The *attribute value* cannot be the same as that of any other unique identifier attribute in the same "ID" environment.

4. If an *attribute value* has been defined by an *attribute declaration* as an *id reference value*:

   a. Its *attribute value* must conform to the model: *name*, (*s*+, *name*)*

   b. The uppercase form of the *attribute value* is used, regardless of how it was entered.

   c. Each *name* in the *attribute value* must be the same as the *attribute value* of a unique identifier attribute in the same "ID" environment, or in an "ID" environment that occurs within it.

5. The ***avo*** and ***avc*** can be omitted only if the *attribute value* contains only ***NAMECHAR*** characters. Either both delimiters must be omitted, or neither.

4.4.2.1 Attribute Value

[14] attribute value = ***CSDATA*** | (***UCSDATA*** | ***BITDATA*** | *character* | *element* | *entity reference* | *parameter entity reference*)*

Notes:

1. An element's *attribute declaration* determines whether ***CSDATA***, ***BITDATA***, ***UCSDATA***, *character*, and *element* are allowed in the *attribute value* structurally, which affects (but does not of itself fully specify) what can occur in the *attribute value* syntactically:

   a. An *entity reference* or *parameter entity reference* can occur whenever ***CSDATA*** is not allowed structurally.

   b. If neither ***CSDATA*** nor *character* is allowed structurally, a *character* can occur only as part of an *element*, *entity reference*, or *parameter entity reference*, except that an extraneous ***SPACE***, ***RS***, ***RE***, or ***NR*** can occur, but is ignored.

   c. If *character* is allowed structurally, a *character* that is not part of an *element* (if allowed) or *other markup* is data. A data *character* can also be part of a data tag for an *element* with "D" minimization.

   d. If ***CSDATA*** is allowed structurally, a *character* can occur only as data.

2. If a data *character* is part of the sequence "***RE***, ***RS***" it will be ignored if the sequence immediately follows the ***avo*** or if it immediately precedes the ***avc***.

3. If an *attribute value* has been defined by an *attribute declaration* to contain one or more names, the uppercase form of each declared *name* is used, regardless of how it was entered.

4. An *entity reference* or *parameter entity reference* is replaced in the *attribute value* by its entity. The result must be syntactically valid and must comply with the *attribute declaration*.

5. The length of an *attribute value* cannot exceed the "PARMLEN" value specified on the *SGML specification*.

4.4.2.2 Name

[15] name = ***NAMESTRT***, ***NAMECHAR****

where:

NAMESTRT is one of the subset of ***NAMECHAR*** characters that is allowed to begin a *name*.

Notes:

1. The ***NAMESTRT*** characters are assigned by the *character use* parameter of the *SGML specification*.
2. The uppercase form of each character, as specified by the "NAMECASE" keyword of the *SGML specification*, is substituted for the bit combinations actually entered.
3. The length of a *name* must not exceed the "NAMELEN" value of the *SGML specification*.

4.4.2.3 Rank Number

[16] rank number = (***plus*** | ***minus***)?, *numeral**

[17] numeral = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

Notes:

1. If *numeral* is omitted when ***plus*** or ***minus*** is present, a *numeral* of "1" is implied.

4.4.2.4 Separator

[18] s = ***SPACE*** | (***RE***, ***RS***) | ***NR***

4.5 Entity Reference

[19] entity reference = ***ermo***, *name*, ***erc***?

Notes:

1. An *entity reference* is valid only if there was an *entity declaration* for the *name*.
2. The ***erc*** can be omitted only it is followed by an *s*.

4.6 Parameter Entity Reference

[20] parameter entity reference = ***ermo***, *parameter entity name*, ***erc***?

[21] parameter entity name = ***PENTCHAR***, ***NAMECHAR***+

Notes:

1. A *parameter entity reference* is valid only if there was an *entity declaration* for the *parameter entity name*.

2. The uppercase form of each ***NAMECHAR*** character, as specified by the "NAMECASE" keyword of the *SGML specification*, is substituted for the bit combinations actually entered.

3. The length of a *parameter entity name* must not exceed the "NAMELEN" value of the *SGML specification*.

4. The ***erc*** can be omitted only if it is followed by an *s*.

4.7 Self-delimiting Entity Reference

[22] self-delimiting entity reference = *character*+

Notes:

1. A *character* string is considered a *self-delimiting entity reference* only if it was defined as such by an *entity declaration*.

4.8 Processing Instruction

[23] processing instruction = ***pio***, ***SYSDATA***, ***pic***

Notes:

1. ***SYSDATA*** is not specified by this Standard. The system must define the means of determining when the data ends.

4.9 Markup Declaration

[24] markup declaration = ***mdo***, (*attribute declaration* | *delimiter declaration* | *entity declaration* | *structure declaration*)?, *ps*?, ***mdc***

[25] ps = *s*+, (***pso***, ***CSDATA***, ***psc***, *s*+)*

Notes:

1. A markup declaration takes effect immediately after it ends.

2. A *ps* is a parameter separator. The ***CSDATA*** in a *ps* is a comment.

4.9.1 Parameter

A *markup declaration* is composed of a "declaration name" and one or more "parameters." In the syntax productions for declarations, the initial syntactic literal is the name (e.g., "ENTITY") and whatever occurs between a *ps* and the next (required or optional) *ps* is a parameter.

A parameter that the syntax productions ultimately define to be a syntactic literal (e.g., "IGNORE") or a delimited syntactic literal (e.g., *hexadecimal literal*) is called a "keyword." One that is ultimately defined to be a syntactic terminal (e.g., *name*) or a delimited syntactic terminal (:e.g., *character literal*) is called a "variable."

Parameters occur in a *marked section start* and in an *SGML specification* as well as in a *markup declaration*.

Notes:

1. The length of a parameter cannot exceed the "PARMLEN" value of the *SGML specification*.
2. Letters in keyword parameters and declaration names are shown in uppercase in the formal specification. The lowercase letters can also be used if the "NAMECASE" parameter of the *SGML specification* includes them and specifies the uppercase letters as their corresponding uppercase forms.
3. Unless otherwise specified, the ***mdo***, ***mdc***, declaration name, parameter separators, and keyword parameters must be entered directly, but a variable parameter can be entered with a *parameter entity reference*:
   a. Each *parameter entity reference* is replaced by its entity.
   b. The variable must still be valid after replacement.
   c. If the variable must begin or end with a delimiter or sequence of delimiters, the required delimiters can be entered explicitly if they are not included in the entity.

4.9.2 Entity Declaration

[26] entity declaration = "ENTITY", *ps*, *entity name*, *ps*, *entity*, (*ps*, "XPI")?

An *entity declaration* associates a name with an entity, or cancels an association.

The "XPI" keyword is used when the system requires that an immediately executable *processing instruction* follow the declaration to specify the entity's storage location or other information.

4.9.2.1 Entity Name

[27] entity name = *name* | *literal* | *parameter entity name* | (***grpo***, *s**, (*name* | *literal* | *parameter entity name*), (*s**, ***or***, *s**, (*name* | *literal* | *parameter entity name*))*, *s**, ***grpc***)

The *name* of the entity. If it is a *literal*, it is a self-delimiting reference. If it is a group, an entity is declared for each *name*, *parameter entity name*, or *literal* in the group.

Notes:

1. A *literal* is allowed only if "SDENTITY YES" is specified on the *SGML specification*.
2. The length of a *literal* and the number of self-delimiting entities that can exist concurrently cannot exceed the values specified on the *SGML specification*.

3. The *literal* is subject to the same selection rules as if it were being assigned to a "class 1, code D2" delimiter role.

4. The following parameter entities are declared by the system and can be referenced only where expressly permitted:

Name	**Requirement**	**Meaning**
#BITDATA	**BITDATA**	***BITDATA***
#CHAR		*character*
#CSDATA		***CSDATA***
#ID	**ID**	unique identifier attribute: *name*
#IDREF	**ID**	unique identifier reference attribute, single reference: *name*
#IDREFS	**ID**	unique identifier reference attribute, one or more references: (*name*, (*s*+, *name*)*)
#NAME		*name*
#NAMES		(*name*, (*s*+, *name*)*)
#NR	**RECORD**	***NR***
#NUM		*decimal number*
#NUMS		(*decimal number*, (*s*+, *decimal number*)*)
#RE	**RECORD**	***RE***
#RS	**RECORD**	***RS***
#S		*s*
#UCSDATA	**UCSDATA**	***UCSDATA***

where:

Name is the *parameter entity name*. It is reserved and cannot be declared explicitly.

Requirement is an *SGML specification* parameter that must be specified as other than "NO" for the parameter entity to be declared.

Meaning is the syntactic construct represented by the parameter entity.

4.9.2.2 Entity

[28] entity = *literal sequence* | "MS" | "PI" | "EXTERNAL" | "CANCEL"

[29] literal sequence = *literal* | (***grpo***, s*, *literal*, (s*, ***seq***, s*, *literal*)*, s*, ***grpc***)

If *entity* is a *literal sequence*, the literals are concatenated (if there are more than one) and stored as the entity.

The meaning of the keywords is as follows:

MS The *marked section* that begins immediately after the *entity declaration* (or after its associated executable *processing instruction*, if any) is stored as the entity.

PI The *processing instruction* that begins immediately after the *entity declaration* (or after its associated executable *processing instruction*, if any) is stored as the entity.

EXTERNAL The entity exists outside of the document source.

CANCEL Existing declarations for the specified entity names are canceled.

Notes:

1. A *character literal* is treated as ***CSDATA*** while it is being stored as an entity; the ***clc*** is the only delimiter recognized.

2. During storage of a *marked section* as an entity:

   a. The parameters of its *marked section start* are ignored.

   b. Delimiter roles in its *content* are disabled, except for those that delimit a nested *marked section*.

   c. A nested *marked section* within the *content* is recognized but its *marked section start* and *marked section end* are treated as data.

3. A *processing instruction* is not executed while it is being stored.

4. The length of a *marked section* or *processing instruction* being stored cannot exceed the "MSENTLEN" quantity of the *SGML specification*.

4.9.2.3 Literal

[30] literal = *character literal* | *decimal literal* | *hexadecimal literal* | *octal literal*

[31] character literal = ***clo***, *character**, ***clc***

[32] decimal literal = ***dec***, ***nld***, *decimal number*, (***nld***, *decimal number*)*

[33] decimal number = *numeral*+

[34] hexadecimal literal = ***hex***, ***nld***, *hexadecimal number*, (***nld***?, *hexadecimal number*)*

[35] hexadecimal number = (*numeral* | "A" | "B" | "C" | "D" | "E" | "F"), (*numeral* | "A" | "B" | "C" | "D" | "E" | "F")

[36] octal literal = ***oct***, ***nld***, *octal number*, (***nld***?, *octal number*)*

[37] octal number = ("0" | "1" | "2"), ("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7"), ("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7")

A *character literal* represents the bit combinations that the document's character set maps to the characters in the literal.

A *decimal literal*, *hexadecimal literal*, or *octal literal* represents the bit combinations that are the numeric equivalents of the numbers in the literal.

Notes:

1. A reference to an #RS, #RE, or #NR entity can occur in a *literal*. It is replaced with the correct character or number base representation.

4.9.3 Structure Declaration

[38] structure declaration = "STRUC", (*ps*, *coexistent structure*)?, (*ps*, *element name*, (*ps*, *minimization*)?, *ps*, *declared content*, (*ps*, *model exceptions*)?)+

A *structure declaration* defines the content and markup minimization allowed for one or more specified element types. The order in which elements and characters occur within such an element in the document must comply with the declaration.

A *structure declaration* cancels any existing structure or attribute declarations for the element types to which it applies.

4.9.3.1 Coexistent Structure

[39] coexistent structure = "COSTRUC", *ps*, *decimal number*

The coexistent structure to which the declaration applies. If omitted, the declaration applies to the base structure.

Notes:

1. This parameter can be specified only if "COSTRUC YES" is specified on the *SGML specification*.

4.9.3.2 Element Type

[40] element type = *name* | *ranked element* | *ranked group* | (**grpo**, *s**, (*name* | *ranked element* | *ranked group*), (*s**, **or**, *s**, (*name* | *ranked element* | *ranked group*))*, *s**, **grpc**)

[41] ranked element = *name*, **ranko**, *rank number*, (**range**, *rank number*?)?, **rankc**

[42] ranked group = *name group*, **ranko**, *rank number*, (**range**, *rank number*?)?, **rankc**

The generic identifier of the type of element to which the declaration applies. If it is a group, the declaration applies to each *name* in the group.

If it is a *ranked group*, the ranks of the group members are kept equal: when the rank of one changes, the ranks of the others are changed to match it.

Notes:

1. A *ranked element* or *ranked group* can be specified only if "RANK NO" is not specified on the *SGML specification*.

4.9.3.3 Markup Minimization

[43] minimization = *start-tag minimization*, *ps*, *end-tag minimization*

[44] start-tag minimization = ("D"? @ "N"? @ "O"?) | **less**

[45] end-tag minimization = *start-tag minimization* | ("C"? @ "D"? @ "E"? @ "N"? @ "O"?) | **less**

where:

O means the *start-tag* is omittable when the *content model* of the element containing this one permits only this type of element to occur next and that the *end-tag* is omittable when the *content model* for this element:

1. does not allow the following element in this element's *content*; or
2. specifies that this element must have a null content.

E means the *end-tag* is omittable when the element's *content* is followed by the *end-tag* of an element containing this one whose *structure declaration* specifies "C" minimization.

N means null tags are permitted.

D means data characters in the element containing this one can be interpreted as this element's tag ("data tag") if they conform to a *literal model* that precedes (for *start-tag minimization*) or follows (for *end-tag minimization*) this element's GI in the containing element's *content model.*

Notes:

1. If either *start-tag minimization* or *end-tag minimization* is specified, both must be. If no minimization is applicable to one of them, the ***less*** delimiter must be specified for it.
2. "O" is allowed only if "OMITTAG YES" is specified on the *SGML specification.*
3. "C" and "E" are allowed only if "CENDTAG YES" is specified on the *SGML specification.*
4. "D" is allowed only if:
   a. "DATATAG YES" is specified on the *SGML specification*; and
   b. The *content model* for the element containing this one includes a *literal model* preceding this element's GI (for *start-tag minimization*) or following it (for *end-tag minimization*) and only a ***seq*** connector separates the *literal model* from the GI.
5. "N" is allowed only if "NULLTAG YES" is specified on the *SGML specification.*
6. If the *declared content* contains a *csdata reference*, neither "D" nor "O" end-tag minimization is allowed.

4.9.3.4 Declared Content

[46] declared content = (***grpo***, *s**, *csdata reference*?, *s**, ***grpc***) | *content model*

[47] csdata reference = ***ero***, *"#CSDATA"*, ***erc****?*

Notes:

1. The ***erc*** *can be omitted only if it is followed by an s.*

4.9.3.5 Content Model

[48] content model = ***grpo***, *s**, (*content token* | *content model*), (*s**, *connector*, *s**, (*content token* | *content model*))*, *s**, ***grpc***, *occurrence indicator*?

[49] content token = (*name* | *character reference* | *literal token* | (***ero***, ("#BITDATA" | "#UCSDATA"), ***erc***?) | *ranked element*), *occurrence indicator*?

[50] character reference = ***ero***, "#CHAR", ***erc***?

[51] literal token = *literal model* | *name reference* | *number reference* | *literal* | (**ero**, "#S", **erc**?)

[52] name reference = **ero**, ("#NAME" | "#NAMES"), **erc**?

[53] number reference = **ero**, ("#NUM" | "#NUMS"), **erc**?

[54] literal model = **grpo**, *s**, (*literal*, *occurrence indicator*?), (*s**, *connector*, *s**, (*literal*, *occurrence indicator*?))*, *s**, **grpc**, *occurrence indicator*?

[55] connector = **and** | **or** | **seq**

[56] occurrence indicator = **opt** | **orep** | **rep**

A *content model* specifies the order and number of occurrences of elements and literal characters.

Connectors specify the ordering:

seq	One of each, in the order entered.
and	One of each, in any order.
or	One only.

Occurrence indicators specify the number of occurrences:

opt	Optional (0 or 1 time).
rep	Required and repeatable (1 or more times).
orep	Optional and repeatable (0 or more times).

Notes:

1. Only one kind of *connector* can occur in a single *content model* (but a *content model* nested within it could have a different *connector*).

2. Only one kind of *connector* can occur in a single *literal model*.

3. A *content token* can be a *literal token* only if "DATATAG YES" is specified on the *SGML specification*.

4. A *content token* can be a *ranked element* only if "RANK NO" is not specified on the *SGML specification*.

5. A *content token* can be a reference to #BITDATA or #UCSDATA only if "BITDATA YES" or "UCSDATA YES", respectively, is specified on the *SGML specification*.

6. An **erc** can be omitted only if it is followed by an *s*.

4.9.3.6 Content Exceptions

[57] content exceptions = **grpo**, *s**, *content exception token*, (*s**, *or*, *s**, *content exception token*)*, *s**, **grpc**

[58] content exception token = (**with** | **less**), (*name* | *ranked element* | *literal*)

The *content exceptions* are elements or literals that are allowed (if prefixed with ***with***) or disallowed (if prefixed with ***less***) anywhere within the bounds of the content defined by the *content model*, including the content of subelements.

If the *content model* is null (nothing between ***grpo*** and ***grpc***), the *content exceptions* are names of attributes that must be specified in a document (if prefixed with ***with***) or not specified (if prefixed with ***less***) for the null *content model* to apply. If it does not apply, the parameters that follow in the *structure declaration* define an alternative content that applies instead.

Notes:

1. If the *content model* is null, ranked elements and literals are not allowed in a *content exception token*.
2. A *ranked element* can be specified only if "RANK NO" is not specified on the *SGML specification*.

4.9.4 Attribute Declaration

[59] attribute declaration = "ATT", (*ps*, *coexistent structure*)?, *ps*, *element type*, (*ps*, *name group*, ((*ps*, "A")?, *ps*, *declared value*, (*ps*, *value exceptions*)?)?, *ps*, *default*)+

An *attribute declaration* defines the attributes of an element type, their allowed values, and their default values. A *structure declaration* must already exist for the element.

The *coexistent structure* and *element type* parameters are the same as for a *structure declaration*.

4.9.4.1 Name Group

[60] name group = *name* | (***grpo***, *s**, *name*, (*s**, ***or***, *s**, *name*)*, *s**, ***grpc***)

The name of the attribute. If it is a group, the declaration applies to each attribute *name* in the group.

Notes:

1. A *name* cannot be the same as a *name token* in the same *attribute declaration*.

4.9.4.2 Markup Minimization

If "A" is specified, the attribute's name and the ***vi*** can be omitted from the document.

Notes:

1. The "A" can be specified only if "ATTMIN YES" is specified on the *SGML specification* and the *declared value* is a *minimizable value*.

4.9.4.3 Declared Value

[61] declared value = *basic value* | *value model*

[62] basic value = *id value* | *id reference value* | (***grpo***, *s**, (*minimizable value* | *name reference* | *number reference* | (*character reference*, *occurrence indicator*?) | *csdata reference*), *s**, ***grpc***)

[63] id value = ***grpo***, *s**, ***ero***, "#ID", ***erc***?, *s**, ***grpc***

[64] id reference value = ***grpo***, *s**, ***ero***, ("#IDREF" | "#IDREFS"), ***erc***?, *s**, ***grpc***

[65] minimizable value = *name token*, (*s**, ***or***, *s**, *name token*)*

[66] name token = ***NAMECHAR***+

Notes:

1. If "ATTEXT NO" is specified on the *SGML specification*, a *declared value* must be a *basic value*.
2. Only one *id value* is allowed in an *attribute declaration*.
3. A *name token*:
   a. Cannot exceed the "NAMELEN" value specified on the *SGML specification*.
   b. Must not be the same as any other *name token* in the same declaration.
   c. Must not be the same as an attribute *name* in the same declaration.

4.9.4.4 Value Model

[67] value model = ***grpo***, *s**, (*value token* | *value model*), (*s**, *connector*, *s**, (*value token* | *value model*))*, *s**, ***grpc***, *occurrence indicator*?

[68] value token = (*name* | *character reference* | *literal token* | (***ero***, ("#BITDATA" | "#UCSDATA"), ***erc***?)), *occurrence indicator*?

Notes:

1. If a *value token* is an element *name*, there must be a *structure declaration* for the element.
2. Only one kind of *connector* can occur in a single *value model* (but a *value model* nested within it could have a different *connector*).
3. A *value token* can be a *literal token* only if "DATATAG YES" is specified on the *SGML specification*.
4. A *value token* can be a reference to #BITDATA or #UCSDATA only if "BITDATA YES" or "UCSDATA YES", respectively, is specified on the *SGML specification*.

4.9.4.5 Value Exceptions

[69] value exceptions = ***grpo***, *s**, *value exception token*, (*s**, *or*, *s**, *value exception token*)*, *s**, ***grpc***

[70] value exception token = (***with*** | ***less***), (*name* | *literal*)

The *value exceptions* are elements or literals that are allowed (if prefixed with ***with***) or disallowed (if prefixed with ***less***) anywhere within the bounds of the *attribute value* defined by the *value model*, including the content of subelements.

Notes:

1. A *value exception token* can be specified only if "ATTEXT YES" is specified on the *SGML specification*.

2. A *value exception token* can be a *literal* only if "DATATAG YES" is specified on the *SGML specification*.

4.9.4.6 Default Value

[71] default = *literal* | "REQUIRED" | "UNKNOWN"

A *literal* is the default *attribute value* that will be used if the attribute is not specified in the document.

If the *default* is "REQUIRED", the attribute must be specified in the document.

If the *default* is "UNKNOWN", the user must define the means of determining the *attribute value* when there is no *attribute specification* in the document.

A short form of an *attribute declaration*, in which the "A" minimization, *value declaration*, and *value exceptions* are omitted, can be used when the sole purpose is to change the default values of already declared attributes. Only the specified attributes are affected; other attributes retain their existing defaults.

Notes:

1. "REQUIRED" cannot be specified for an attribute of an element whose *structure declaration* specifies "O" or "D" start tag minimization.

2. If the short form is used for one attribute, it must be used for all attributes in the declaration.

4.9.5 Delimiter Declaration

[72] delimiter declaration = "DELIM", (*ps*, ("ADD" | "DEL" | "REP") , (*ps*, *role*, (*ps*, *literal group*)?)+)+

[73] literal group = *literal* | (***grpo***, *s**, *literal*, (*s**, ***or***, *s**, *literal*)*, *s**, ***grpc***)

A *delimiter declaration* changes the character strings currently assigned to delimiter roles.

4.9.5.1 Function

The keywords mean:

ADD The character strings in the *literal group* are added to those assigned to the specified delimiter *role*.

DEL The character strings in the *literal group* are deleted from those assigned to the specified delimiter *role*. If the group is empty or has only null strings, all the current assignments are deleted, which disables the delimiter.

REP The character strings in the *literal group* replace those assigned to the specified delimiter *role*. If the group is empty or has only null strings, the current assignments are replaced with those that existed at the end of the *prolog*, or at the start of the current delimiter environment if one was defined by a *marked section*.

4.9.5.2 Roles and Assigned Strings

The delimiter roles are classified by use for the purpose of applying selection rules for the character strings assigned to the roles. The classes are:

Class	Roles	Where Roles are Used
0	**ERO**	Anywhere
1	**CNET CnEE CnES EEMO ESMO MDO NET NST PIO**	Element content (and Class 3 with ATTEXT feature)
2	**VI**	Descriptive markup after a *name*
3	**AVC**	Attribute value (and Class 2 with ATTEXT feature)
4	**ESMC**	Start-tag
5	**EEMC**	End-tag
6	**AVO**	Descriptive markup after a ***vi***
7	**ERC**	Entity reference
8	**PIC**	Processing instruction
9	**OPT OREP RANKO REP**	Suffix
10	**AND CLO DEC GRPC GRPO HEX MDC OCT OR PSO SEQ**	Parameter start or separator
11	**LESS WITH**	Prefix
12	**NLD**	Numeric literal
13	**MINUS PLUS RANGE RANKC**	Ranked element or ranked group
14	**CLC**	Character literal
15	**PSC**	Parameter separator comment

The rules for selecting strings to be assigned to delimiter roles are as follows:

1. The same string cannot be assigned to more than one *role* in the same class.

2. The characters and pseudocharacters can be included in strings assigned to the roles as follows:

BITCOM	Any role.
NAMECHAR	(Other than ***NAMESTRT***) No roles.
NAMESTRT	No roles except ***DEC***, ***HEX***, and ***OCT***.
NR, RE, RS	Roles in classes 1, 3, 14, and 15.
PENTCHAR	Any role not in class 10.
SPACE	Roles in classes 1, 3, 14, and 15.
SYSCHAR	No roles.

3. There are additional criteria indicated by the following list of delimiter roles. For each *role* the list states:

String The character string that is assigned to the delimiter if no *delimiter declaration* is issued and the *role* is not specified on the *SGML specification delimiter use* parameter.

Class The class to which the *role* is assigned.

Codes One or more coded designations for additional string selection criteria. The codes are explained at the end of the list.

Description The function of the delimiter.

Role	String	Class	Codes	Description
AND	**@**	**10**	**M2**	And connector
AVC	**"**	**3**	**M3**	Attribute value close
AVO	**"**	**6**	**D3**	Attribute value open
CLC	**"**	**14**		Character literal close
CLO	**"**	**10**	**M2**	Character literal open
CNET	**|**	**1**	**D2, F2, M1**	Conditional null end-tag
C*n*EE	**#*n***	**1**	**C1, F1, M1**	Coexistent structure, element end ("*n*"is a *decimal number*)
C*n*ES	**@*n***	**1**	**C1, F1, M1**	Coexistent structure, element start ("*n*"is a *decimal number*)
DEC	**D**	**10**	**C5, M2**	Decimal numeric literal
EEMC	**>**	**5**	**D2**	Element end, markup close
EEMO	**</**	**1**	**C3, M1**	Element end, markup open
ERC	**.**	**7**	**D2**	Entity reference close
ERO	**&**	**0**	**C4, M4**	Entity reference open
ESMC	**>**	**4**	**D2**	Element start, markup close
ESMO	**<**	**1**	**C2, M1**	Element start, markup open
GRPC	**)**	**10**	**M2**	Group close
GRPO	**(**	**10**	**M2**	Group open
HEX	**H**	**10**	**C5, M2**	Hexadecimal numeric literal
LESS	**-**	**11**	**M2**	Exclusion
MDC	**>**	**10**	**D2, M2**	Markup declaration close
MDO	**<!**	**1**	**M1**	Markup declaration open
MINUS	**-**	**13**	**F3**	Rank number decrement
NET	**>**	**1**	**D2, F2, M1**	Null end-tag
NLD	**/**	**12**		Numeric literal divider
NST	**%**	**1**	**D2, F2, M1**	Null start-tag
OCT	**O**	**10**	**C5, M2**	Octal numeric literal
OPT	**?**	**9**	**M2**	Optional
OR	**|**	**10**	**M2**	Or connector
OREP	*****	**9**	**M2**	Optional and repeatable
PIC	**>**	**8**	**D2**	Processing instruction close
PIO	**<?**	**1**	**D1, M1**	Processing instruction open
PLUS	**+**	**13**	**F3**	Rank number increment
PSC	**--**	**15**		Parameter separator close
PSO	**--**	**10**	**M2**	Parameter separator open
RANGE	**...**	**13**	**F3**	Rank range
RANKC	**]**	**13**	**F3**	Rank close
RANKO	**[**	**9**	**F3, M2**	Rank open
REP	**+**	**9**	**M2**	Repeatable
SEQ	**,**	**10**	**M2**	Sequence connector
VI	**=**	**2**	**M3**	Value indicator
WITH	**+**	**11**	**M2**	Inclusion

The codes have the following meaning:

C Contextual Delimiters

The delimiter roles with codes C1 through C5 are context dependent and are recognized as delimiters only when followed by the character indicated by the code:

C1 ***NAMESTRT***
C2 ***NAMESTRT*** or, if "NULLTAG YES" is specified on the *SGML specification*, ***esmc***
C3 ***NAMESTRT*** or, if "NULLTAG YES" is specified on the *SGML specification*, ***eemc***
C4 ***NAMESTRT*** or ***PENTCHAR***
C5 ***nld***

The sequence composed of a contextual delimiter followed by its required character must differ from other such sequences and delimiters in the same class.

D Data Leaders

Codes D1 through D3 identify delimiters that can be followed by unparsable data under the conditions indicated by the codes:

D1 Always
D2 If "BITDATA YES" or "UCSDATA YES" is specified on the *SGML specification*.
D3 If "ATTEXT YES" and either "BITDATA YES" or "UCSDATA YES" are specified on the *SGML specification*.

When the conditions apply, a data leader delimiter cannot be a substring of another delimiter in the same class.

F Feature Dependencies

The roles with codes F1 through F3 can have strings assigned only if the *feature use* indicated by the code is specified on the *SGML specification*:

F1 "COSTRUC YES"
F2 "NULLTAG YES"
F3 "RANK YES" or "RANK ENVIRON"

M Multiclass Relationships

Interrelationships among classes are indicated by the codes:

M1 When applying delimiter string selection criteria, a *self-delimiting entity reference* is considered a delimiter role in class 1. If "ATTEXT YES" is specified on the *SGML specification*, the class 1 roles are also part of class 3.
M2 A character string assigned to a *role* in class 9 or 11 cannot duplicate a string assigned to a *role* in class 10.
M3 If "ATTEXT YES" is specified on the *SGML specification*, the ***avc*** is also in classes 1 and 2.
M4 A string assigned to the ***ero*** cannot duplicate a string assigned to any *role* in any class.

Notes:

1. A *role* cannot be specified if it was specified as "FIXED" on the *SGML specification delimiter use* parameter.

2. Unless otherwise specified on the *SGML specification delimiter use* parameter:

   a. The maximum length of a delimiter string for a *role* is the length of the initial string assigned to it in the Standard. For the ***cnee*** and ***cnes***, it is 2 characters.

b. The number of strings that can be assigned concurrently to the same delimiter *role* is 1.

4.10 Marked Section

[74] marked section = *marked section start*, *content*, *marked section end*

[75] marked section start = "MS", (*ps*, "TEMP")?, (*ps*, "IGNORE")?, (*ps*, ("DATA" | *new environments*))?

[76] marked section end = ***mdo***, "MSE", *ps*?, ***mdc***

Notes:

1. The *marked section end* must be in the same entity as the *marked section start*.
2. "IGNORE" and *new environments* can be specified with an *entity reference* or *parameter entity reference*.
3. The applicable *structure declaration* for the *element* in which the *marked section* occurs when processed determines what is allowed in the section's *content*.

4.10.1 Keyword Parameters

"TEMP" identifies the section as a temporary part of the document that might need to be removed at a later time.

"IGNORE" specifies that the section is treated as having a null *content*.

"DATA" specifies that the *content* is treated as data.

Notes:

1. If "IGNORE" or "DATA" is specified, a nested *marked section* within the *content* is recognized but its *marked section start* and *marked section end* are treated as data. The *marked section* is terminated by the first *marked section end* that is not part of a nested *marked section*

4.10.2 Markup Environments

[77] new environments = ***grpo***, *s**, *environment*, (*s**, ***and***, *s**, *environment*)*, *s**, ***grpc***)

The marked section constitutes a new markup environment of the type (or types) specified. Only *environment* names from the following list are allowed:

DELIM	The effect of any *delimiter declaration* issued within the section is canceled at the end.
ENTITY	The effect of any *entity declaration* issued between the end of the *prolog* and the start of the section is canceled at the start of the section and restored at the end. The effect of any *entity declaration* issued within the section is canceled at the end.
ID	Unique identifiers defined within the section are maintained independently of those outside of it, and can therefore duplicate them. Resolution of ID references within the

	section is attempted first with IDs defined in the section, then with IDs defined in "ID environment" marked sections nested within the section, in the order the nested sections occur in the document.
RANK	The ranks that ranked elements had at the start of the section (the "starting ranks") are restored at the end. Ranks within the marked section are added to the starting ranks.
STRUC	The effect of any *structure declaration* or long form of *attribute declaration* issued between the end of the *prolog* and the start of the section is canceled at the start of the section and restored at the end. The effect of any such declaration issued within the section is canceled at the end.

Notes:

1. An *environment* name can be specified only if "ENVIRON" was specified for the similarly named keyword of the *SGML specification feature use* parameter.

4.11 SGML Specification

[78] SGML specification = "SGML", *ps*, *feature use*, *ps*, *variations*

An *SGML specification* specifies the document's use of Standard Generalized Markup Language features and variations of basic functions.

4.11.1 Feature Use

[79] feature use = "FEATURE", *ps*, "ATTEXT", *ps*, ("NO" | "YES"), *ps*, "ATTMIN", *ps*, ("NO" | "YES"), *ps*, "CENDTAG", *ps*, ("NO" | "YES"), *ps*, "COSTRUC", *ps*, ("NO" | "YES"), *ps*, "DATATAG", *ps*, ("NO" | "YES"), *ps*, "DELIM",*ps*, ("FIXED" | "PROLOG" | "CONTENT" | "ENVIRON"), *ps*, "ENTITY", *ps*, ("PROLOG" | "CONTENT" | "ENVIRON"), *ps*, "ID", *ps*, ("NO" | "YES" | "ENVIRON"), *ps*, "NULLTAG", *ps*, ("NO" | "YES"), *ps*, "OMITTAG", *ps*, ("NO" | "YES"), *ps*, "RANK", *ps*, ("NO" | "YES" | "ENVIRON"), *ps*, "RECORD", *ps*, ("NO" | "YES"), *ps*, "SDENTITY", *ps*, ("NO" | ("YES", *ps*, *decimal number*, *ps*, *decimal number*)), *ps*, "STRUC", *ps*, ("PROLOG" | "CONTENT" | "ENVIRON")

where:

CONTENT	means that a declaration of that kind can occur in the *content* of an element and in the *prolog*.
ENVIRON	means that a declaration of that kind can occur in the *content* of an element and in the *prolog*, and that an environment of that type can be specified on the *new environments* parameter of a *marked section start*.
FIXED	means that a *delimiter declaration* is not valid in the *document*.
NO	means the feature is not used.
PROLOG	means that a declaration of that kind can occur only in the *prolog*.

YES means the feature is used.

If "SDENTITY YES" is specified, the following decimal numbers are the maximum length of a *self-delimiting entity reference* and the maximum number of them that can be defined concurrently, respectively.

Notes:

1. The "STRUC" feature affects both structure declarations and the long form of attribute declarations in which a *declared value* is specified.

4.11.2 Variations

[80] variations = *ps*, *quantity*, *ps*, *character use*, *ps*, *delimiter use*, *ps*, *data use*, *ps*, *alias use*

4.11.2.1 Quantitative Characteristics

[81] quantity = "QUANTITY", *ps*, "DESCLEN", *ps*, *decimal number*, *ps*, "MSENTLEN", *ps*, *decimal number*, *ps*, "NAMELEN", *ps*, *decimal number*, *ps*, "PARMLEN", *ps*, *decimal number*

4.11.2.2 Character Use

[82] character use = "CHAR", *ps*, "SYSCHAR", *ps*, *literal sequence*, *ps*, "SPACE", *ps*, *literal sequence*, *ps*, "PENTCHAR", *ps*, *literal sequence*, *ps*, "NAMECHAR", *ps*, *literal sequence*, *ps*, "NAMESTRT", *ps*, *literal sequence*, *ps*, "NAMECASE", *ps*, *literal sequence*

For ***SYSCHAR***, ***SPACE***, ***PENTCHAR***, and ***NAMECHAR*** the bit combinations in the *literal sequence* are assigned to the specified category of the document's character set.

For ***NAMECASE*** the bit combinations are taken in pairs: the first is a normal character in ***NAMECHAR***; the second is the associated uppercase form, which must also be in ***NAMECHAR*** and can be the same as the first.

For ***NAMESTRT*** the bit combinations must be in ***NAMECHAR***.

Notes:

1. The ***SYSCHAR***, ***SPACE***, ***PENTCHAR***, and ***NAMECHAR*** bit combinations must be mutually exclusive.

   a. ***SYSCHAR*** must contain eight or more bit combinations.

   b. ***SPACE*** must include the space character (decimal 32 in ISO 646 IRV).

   c. ***PENTCHAR*** must include the number sign (#; decimal 35 in ISO 646 IRV).

   d. ***NAMECHAR*** must include the numerals "0" through "9" (decimal 48 through 57 in ISO 646 IRV) and the uppercase letters "A" through "Z" (decimal 65 through 90 in ISO 646 IRV). When the document's character set uses a 7-bit or larger code, ***NAMECHAR*** must also include the lowercase letters "a" through "z" (decimal 97 through 122 in ISO 646 IRV).

2. The bit combination for a *character* in the document's *character set* need not be the same as that character's bit combination in ISO 646 IRV.

3. ***NAMESTRT*** must include the uppercase letters and, if they are included in ***NAMECHAR***, the lowercase letters.

4. Each bit combination in ***NAMECHAR*** must occur as the first of a pair in ***NAMECASE***.

4.11.2.3 Delimiter Use

The *delimiter use* parameter defines any variations from the standard delimiter assignments that are used in the document and any restrictions on reassignment.

[83] delimiter use = "DELIM", *ps*, ("STANDARD" | ("CUSTOM", (*ps*, *role*, *ps*, *literal group*, *ps*, ("FIXED" | (*decimal number*, *ps*, *decimal number*)))+))

where:

STANDARD	means the initial delimiter strings, maximum lengths, and maximum counts are those defined in this Standard.
CUSTOM	means that the initial strings, maximum length, and maximum count for one or more roles are specified in this parameter.
FIXED	means that the preceding *role* cannot be specified on a *delimiter declaration*.

The *role* and *literal group* are the same as for a *delimiter declaration*.

The decimal numbers that are specified when "FIXED" is not are the maximum length of a delimiter string and the maximum count of concurrent assignments, respectively.

4.11.2.4 Data Content Characteristics

[84] data use = "DATA", *ps*, "BITDATA", *ps*, ("NO" | "YES"), *ps*, "UCSDATA", *ps*, ("NO" | "YES")

4.11.2.5 Alias Use

[85] alias use = "ALIAS", *ps*, ("NO" | ("YES", (*ps*, *standard name*, *ps*, *name*, (*ps*, *standard keyword*, *ps*, *name*)*)+))

[86] standard name = "ATT" | "DELIM" | "ENTITY" | "MS" | "STRUC"

The *alias use* parameter specifies any replacement in the document for a declaration name, keyword, or system-declared *parameter entity name* defined in the Standard. When no alias *name* is specified, the name defined in the Standard is used.

The *standard name* determines each *standard keyword* that can be specified, in accordance with the following list:

Name	**Standard Keywords**
ATT	A, REQUIRED, UNKNOWN
DELIM	ADD, DEL, REP, *role*

ENTITY	CANCEL, EXTERNAL, MS, PI, XPI, #BITDATA, #CHAR, #CSDATA, #ID, #IDREF, #IDREFS, #NAME, #NAMES, #NR, #NUM, #NUMS, #RE, #RS, #S, #UCSDATA
MS	DATA, DELIM, ENTITY, ID, IGNORE, MSE, RANK, STRUC, TEMP
STRUC	C, COSTRUC, D, E, N, O

Notes:

1. The alias for a *standard name* cannot be another *standard name*, or an alias for one.

2. The alias for a *standard keyword* cannot be another *standard keyword* for the same *standard name*, or an alias for one.

3. The "COSTRUC" keyword is specified for "STRUC" but applies to the *attribute declaration* as well.

4. When *standard name* is "ENTITY", the parameter entity names that can be specified are limited to those declared by the system given the specified *feature use* parameter.

5. When *standard name* is "DELIM", the *role* names that can be specified are those specifiable for a *delimiter declaration* given the specified *feature use* and *delimiter use* parameters.

5 Conformance to the Standard

5.1 Conforming Documents

5.1.1 Basic SGML Document

A document is a "basic SGML document" if it meets the following criteria:

1. It conforms to the clause in this Standard entitled "4 Formal Specification" on page 52.
2. Its *SGML specification*, exclusive of the *character use* parameter, is

```
<!SGML

FEATURE  ATTEXT      NO
         ATTMIN      NO
         CENDTAG     NO
         COSTRUC     NO
         DATATAG     NO
         DELIM       FIXED
         ENTITY      PROLOG
         ID          NO
         NULLTAG     NO
         OMITTAG     NO
         RANK        NO
         RECORD      NO
         SDENTITY    NO
         STRUC       PROLOG

QUANTITY DESCLEN     960
         MSENTLEN    64000
         NAMELEN     8
         PARMLEN     240

DELIM    STANDARD

DATA     BITDATA     NO
         UCSDATA     NO

ALIAS    NO
>
```

3. Either its code exceeds seven bits and its *SGML specification character use* parameter, exclusive of "SYSCHAR," is

```
CHAR      -- SYSCHAR can be any valid specification. --
          SPACE          D/32
          PENTCHAR       "#"
          NAMECHAR      ("ABCDEFGHIJKLMNOPQR",
                         "STUVWXYZ0123456789",
                         "abcdefghijklmnopqr",
                         "stuvwxyz")
          NAMESTRT      ("ABCDEFGHIJKLM",
                         "NOPQRSTUVWXYZ",
                         "abcdefghijklm",
                         "nopqrstuvwxyz")
          NAMECASE      ("AABBCCDDEEFFGGHHII",
                         "JJKKLLMMNNOOPPQQRR",
                         "SSTTUUVVWWXXYYZZ",
                         "00112233445566778899",
                         "aAbBcCdDeEfFgGhHiI",
                         "jJkKlLmMnNoOpPqQrR",
                         "sStTuUvVwWxXyYzZ")
```

or its code is smaller than seven bits and the *character use* parameter is

```
CHAR      -- SYSCHAR can be any valid specification. --
          SPACE          D/32
          PENTCHAR       "#"
          NAMECHAR      ("ABCDEFGHIJKLMNOPQR",
                         "STUVWXYZ0123456789")
          NAMESTRT      ("ABCDEFGHIJKLM",
                         "NOPQRSTUVWXYZ")
          NAMECASE      ("AABBCCDDEEFFGGHHII",
                         "JJKKLLMMNNOOPPQQRR",
                         "SSTTUUVVWWXXYYZZ",
                         "00112233445566778899")
```

5.1.2 Extended SGML Document

A document is an "extended SGML document" if it would satisfy the criteria for a basic SGML document but for differences in its *SGML specification feature use* parameter.

5.1.3 Variant SGML Document

A document is a "variant basic SGML document" or a "variant extended SGML document" if it would satisfy the criteria for a basic or extended SGML document, respectively, but for differences in its *SGML specification variations* parameters.

5.2 Conforming Systems

5.2.1 Basic SGML Processor

A system is a "basic SGML processor" if it can process any basic SGML document whose *SGML specification* "SYSCHAR" parameter includes all of the bit combinations required by the system. A system can require up to eight such bit combinations.

A reference to a basic SGML processor must be qualified by naming the required "SYSCHAR" bit combinations; e.g., "can process any basic SGML document that does not contain a null, line feed, or carriage return." Or, more formally, "can process any basic SGML document whose *SGML specification* 'SYSCHAR' parameter includes the bit combinations decimal 0, 10, and 13."

5.2.2 Extended SGML Processor

A system is an "extended SGML processor" if it is a basic SGML processor that can also process an extended SGML document.

A reference to an extended SGML processor must be qualified by naming the characteristics of the extended SGML documents it can process; e.g., "dynamic entity declarations and all markup minimization features." A qualified reference can be made more formally with *SGML specification* parameters; e.g., "ENTITY CONTENT and the ATTMIN, CENDTAG, DATATAG, OMITTAG, and NULLTAG features."

5.2.3 Customizable SGML Processor

A basic or extended SGML processor is "customizable" if it can also process variant basic or variant extended SGML documents, respectively.

A reference to a customizable SGML processor must specify the range of *SGML specification variations* specifications that it can process.

5.2.4 Variant SGML Processor

A system is a "variant basic SGML processor" or a "variant extended SGML processor" if it would satisfy the criteria for a basic or extended SGML processor, respectively, except that it can only process variant SGML documents.

A reference to a variant SGML processor must specify the *SGML specification variations* specifications that it requires.

Annex A. Introduction to Generalized Markup

A.1 The Markup Process

Text processing and word processing systems typically require users to intersperse additional information in the natural text of the document being processed. This added information, called "markup," serves two purposes:

1. Separating the logical elements of the document; and

2. Specifying the processing functions to be performed on those elements.

Consider how the user of such a system marks up a document. There are three distinct steps, although he may not perceive them as such.

1. He first analyzes the information structure and other attributes of the document; that is, he identifies each meaningful separate element, and characterizes it as a paragraph, heading, ordered list, footnote, or some other element type.

2. He then determines, from memory or a style book, the processing instructions ("controls") that will produce the format desired for that type of element.

3. Finally, he inserts the chosen controls into the text.

Here is how the start of this paper looks when marked up with controls in a typical text processing formatting language:

```
.SK 1
Text processing and word
processing systems typically
require users to intersperse
additional information in
the natural text of the document
being processed.
This added information, called
"markup," serves two purposes:
.TB 4
.OF 4
.SK 1
1.¬Separating the logical
elements of the document; and
.OF 4
.SK 1
2.¬Specifying the processing
functions to be
performed on those elements.
.OF 0
.SK 1
```

© 1981 ACM. Adapted from Goldfarb, Charles F., "A Generalized Approach to Document Markup," *SIGPLAN Notices*, June 1981, by permission of the author and the Association for Computing Machinery.

The .SK, .TB, and .OF controls, respectively, cause the skipping of vertical space, the setting of a tab stop, and the offset, or "hanging indent," style of formatting. (The not sign (¬) in each list item represents a tab code, which would otherwise not be visible.)

Procedural markup like this, however, has a number of disadvantages. For one thing, information about the document's attributes is usually lost. If the user decides, for example, to center both headings and figure captions when formatting, the "center" control will not indicate whether the text on which it operates is a heading or a caption. Therefore, if he wishes to use the document in an information retrieval application, search programs will be unable to distinguish headings—which might be very significant in information content—from the text of anything else that was centered.

Procedural markup is also inflexible. If the user decides to change the style of his document (perhaps because he is using a different output device), he will need to repeat the markup process to reflect the changes. This will prevent him, for example, from producing double-spaced draft copies on an inexpensive computer line printer while still obtaining a high quality finished copy on an expensive photocomposer. And if he wishes to accept competitive bids for the typesetting of his document, he will be restricted to those vendors that use the identical text processing system, unless he is willing to pay the cost of repeating the markup process.

Moreover, markup with control words can be time-consuming, error-prone, and require a high degree of operator training, particularly when complex typographic results are desired. This is true (albeit less so) even when a system allows defined procedures ("macros"), since these must be added to the user's vocabulary of primitive controls. The elegant and powerful TeX system (2), for example, which is widely used for mathematical typesetting, includes some 300 primitive controls and macros in its basic implementation.

These disadvantages of procedural markup are avoided by a markup scheme due to C. F. Goldfarb, E. J. Mosher, and R. A. Lorie (3, 4). It is called "generalized markup" because it does not restrict documents to a single application, formatting style, or processing system. Generalized markup is based on two novel postulates:

1. Markup should describe a document's structure and other attributes rather than specify processing to be performed on it, as descriptive markup need be done only once and will suffice for all future processing.

2. Markup should be rigorous so that the techniques available for processing rigorously-defined objects like programs and data bases can be used for processing documents as well.

These postulates will be developed intuitively by examining the properties of this type of markup.

A.2 Descriptive Markup

With generalized markup, the markup process stops at the first step: the user locates each significant element of the document and marks it with the mnemonic name ("generic identifier") that he feels best characterizes it. The processing system associates the markup with processing instructions in a manner that will be described shortly.

Marked up in the Standard Generalized Markup Language, the start of this paper looks like this:

```
<p>
Text processing and word
processing systems typically
require users to intersperse
additional information in
the natural text of the document
being processed.
This added information, called
<q>markup</q>, serves two purposes:
<ol>
<li>Separating the logical
elements of the document; and
<li>Specifying the processing
functions to be
performed on those elements.
</ol>
```

Each generic identifier (GI) is delimited by a less-than symbol (<) if it is at the start of an element, or by less-than followed by stroke (</) if it is at the end. A greater-than symbol (>) separates a GI from any text that follows it./1/ The mnemonics P, Q, OL, and LI stand, respectively, for the element types paragraph, quotation, ordered list, and list item. The combination of the GI and its delimiters is called a "start-tag" or an "end-tag," depending upon whether it identifies the start or the end of an element.

This example has some interesting properties. There are no quotation marks in the text; the processing for the quotation element generates them and will distinguish between opening and closing quotation marks if the output device permits. The comma that follows the quotation element is not actually part of it but is brought inside the quotation marks during formatting. Similarly, sequence numbers are generated for the ordered list items. The source text, in other words, contains only information; characters whose only role is to enhance the presentation are generated during processing.

If, as postulated, descriptive markup like this suffices for all processing, it must follow that the processing of a document is a function of its attributes. The way text is composed offers intuitive support for this premise. Such techniques as beginning chapters on a new page, italicizing emphasized phrases, and indenting lists, are employed to assist the reader's comprehension by emphasizing the structural attributes of the document and its elements.

From this analysis, a 3-step model of document processing can be constructed:

1. Recognition: An attribute of the document is recognized, e.g., an element with a generic identifier of "footnote."

2. Mapping: The attribute is associated with a processing function. The footnote GI, for example, could be associated with a procedure that prints footnotes at the bottom of the page or one that collects them at the end of the chapter.

3. Processing: The chosen processing function is executed.

Text formatting programs conform to this model. They recognize such elements as words and sentences, primarily by interpreting spaces and punctuation as implicit markup. Mapping is usually via a branch table. Processing for words typically involves determining the word's width and testing for an overdrawn line; processing for sentences might cause space to be inserted between them./2/

---

1 Actually, these characters are just defaults. The Standard Generalized Markup Language permits a choice of delimiter characters.

2 The model need not be reflected in the program architecture; processing of words, for example, could be built into the main recognition loop to improve performance.

In the case of low-level elements such as words and sentences the user is normally given little control over the processing, and almost none over the recognition. Some formatters offer more flexibility with respect to higher-level elements like paragraphs, while those with powerful macro languages can go so far as to support descriptive markup. In terms of the document processing model, the advantage of descriptive markup is that it permits the user to define attributes—and therefore element types—not known to the formatter and to specify the processing for them.

For example, the Standard Generalized Markup Language sample just described includes the element types "ordered list" and "list item," in addition to the more common "paragraph." Built-in recognition and processing of such elements is unlikely. Instead, each will be recognized by its explicit markup and mapped to a procedure associated with it for the particular processing run. Both the procedure itself and the association with a GI would be expressed in the system's macro language. On other processing runs, or at different times in the same run, the association could be changed. The list items, for example, might be numbered in the body of a book but lettered in an appendix.

So far the discussion has addressed only a single attribute, the generic identifier, whose value characterizes an element's semantic role or purpose. Some descriptive markup schemes refer to markup as "generic coding," because the GI is the only attribute they recognize (5). In generic coding schemes, recognition, mapping, and processing can be accomplished all at once by the simple device of using GIs as control procedure names. Different formats can then be obtained from the same markup by invoking a different set of homonymous procedures. This approach is effective enough that one notable implementation, the SCRIBE system, is able to prohibit procedural markup completely (1).

Generic coding is a considerable improvement over procedural markup in practical use, but it is conceptually unsound. Documents are complex objects, and they have other attributes that a markup language must be capable of describing. For example, suppose the user decides that his document is to include elements of a type called "figure" and that it must be possible to refer to individual figures by name. The markup for a particular figure element known as "angelfig" could begin with this start-tag:

```
<fig id=angelfig>
```

"Fig," of course, stands for "figure," the value of the generic identifier attribute. The GI identifies the element as a member of a set of elements having the same role. In contrast, the "unique identifier" (ID) attribute distinguishes the element from all others, even those with the same GI. (It was unnecessary to say "GI=fig," as was done for ID, because in the Standard Generalized Markup Language it is understood that the first piece of markup for an element is the value of its GI).

The GI and ID attributes are termed "primary" because every element can have them. There are also "secondary" attributes that are possessed only by certain element types. For example, if the user wanted some of the figures in his document to contain illustrations to be produced by an artist and added to the processed output, he could define an element type of "artwork." Because the size of the externally-generated artwork would be important, he might define artwork elements to have a secondary attribute, "depth."/3/ This would result in the following start-tag for a piece of artwork 24 picas deep:

```
<artwork depth=24p>
```

The markup for a figure would also have to describe its content. "Content" is, of course, a primary attribute, the one that the secondary attributes of an element describe. The content consists of an

---

[3] "Depth=" is not simply the equivalent of a vertical space control word. Although a full-page composition program could produce the actual space, a galley formatter might print a message instructing the layout artist to leave it. A retrieval program might simply index the figure and ignore the depth entirely.

arrangement of other elements, each of which in turn may have other elements in its content, and so on until further division is impossible./4/ One way in which the Standard Generalized Markup Language differs from generic coding schemes is in the conceptual and notational tools it provides for dealing with this hierarchical structure. These are based on the second generalized markup hypothesis, that markup can be rigorous.

A.3 Rigorous Markup

Assume that the content of the figure "angelfig" consists of two elements, a figure body and a figure caption. The figure body in turn contains an artwork element, while the content of the caption is text characters with no explicit markup. The markup for this figure could look like this:/5/

```
<fig id=angelfig>
<figbody>
<artwork depth=24p>
</artwork>
</figbody>
<figcap>Three Angels Dancing
</figcap>
</fig>
```

The markup rigorously expresses the hierarchy by identifying the beginning and end of each element in classical left list order. No additional information is needed to interpret the structure, and it would be possible to implement support by the simple scheme of macro invocation discussed earlier. The price the user pays for this simplicity, though, is that he must enter an end-tag for every element.

This price is totally unacceptable in practice. The user knows that the start of a paragraph, for example, terminates the previous one, so he will be reluctant to go to the trouble and expense of entering an explicit end-tag for every single paragraph just to share his knowledge with the system. He will have equally strong feelings about other element types he may define for himself, if they occur with any great frequency.

With the Standard Generalized Markup Language, however, it is possible to advise the system about the structure and attributes of any type of element the user creates. This is done by creating a "document type description," using features of the language called structure declarations ("STRUC") and attribute declarations ("ATT"). While the markup in a document consists of descriptions of individual elements, a document type description defines the set of all possible valid markup of a type of element.

For example, the attribute declaration for a figure might look like this. It includes the possible values the attribute can have, and the default value if the attribute is optional and is not specified in the document.

---

4 One can therefore speak of documents and elements almost interchangeably: the document is simply the element that is at the top of the hierarchy for a given processing run. A technical report, for example, could be formatted both as a document in its own right and as an element of a journal.

5 Like "GI=," "content=" can safely be omitted. It is unnecessary when the content is externally generated, it is understood when the content consists solely of tagged elements, and for text characters it is implied by the delimiter (>) that ends the start-tag.

```
<! ATT --
ELEMENT  ATTRIBUTE  VALUE       DEFAULT --
fig      id         (&#ID.)     UNKNOWN  >
<! ATT
artwork  depth      (&#CHAR.+)  REQUIRED >
```

The declaration indicates that a figure can have an ID attribute and that its value must be a unique identifier name. The attribute is optional and does not have a default value if not specified.

In contrast, the depth attribute of the artwork element is required. Its value can be any character string.

A structure declaration includes a "model," expressed in a notation derived from the BNF notation used for formal grammars. Suppose, for example, the user extends his definition of "figure" to permit the figure body to contain either artwork or certain kinds of textual elements. The structure declaration might look like this:/6/

```
<! STRUC
--        ELEMENT   MIN  MODEL                       --
-- 1 --   fig       - C  (figbody, figcap?)
-- 2 --   figbody   - O  (artwork | (p | ol | ul)+)
-- 3 --   artwork   - O  ()
-- 4 --   figcap    - E  (&#CHAR.+)
>
```

Line 1 means that a figure contains a figure body and, optionally, can contain a figure caption following the figure body. (The "C" will be explained shortly.)

Line 2 says the body can contain either artwork or an intermixed collection of paragraphs, ordered lists, and unordered lists. The "O" in the markup minimization field ("MIN") indicates that the element's end-tag can be omitted when it is unambiguously implied by the start of the following element.

Line 3 defines artwork as having a null content, as it will be generated externally and pasted in. As there is no content in the document, there is no need for ending markup, so it can be omitted.

Line 4 defines a figure caption's content as 1 or more characters. A character is a terminal, incapable of further division. The "E" in the "MIN" field indicates the caption's end-tag can be omitted when unambiguously implied by the end-tag of an element that contains the caption and that also has "C" minimization (e.g., "</fig>").

It is assumed that p, ol, and ul have been defined in other structure declarations.

With this formal definition of figure elements available, the following markup for "angelfig" is now acceptable:

```
<fig id=angelfig>
<figbody>
<artwork depth=24p>
<figcap>Three Angels Dancing
</fig>
```

---

[6] The question mark (?) means an element is optional, the comma (,) that it follows the preceding element in sequence, the asterisk (*) that the element can occur 0 or more times, and the plus (+) that it must occur 1 or more times. The vertical bar (|) is used to separate alternatives. Parentheses are used for grouping as in mathematics.

There has been a 40% reduction in markup, since the end-tags for three of the elements are no longer needed.

- As the structure declaration defined the figure caption as part of the content of a figure, terminating the figure automatically terminated the caption.
- Since the figure caption itself is on the same level as the figure body, the <figcap> start-tag implicitly terminated the figure body.
- The artwork element was self-terminating, as the structure declaration defined its content to be null./7/

Document type descriptions have uses in addition to markup minimization./8/ They can be used to validate the markup in a document before going to the expense of processing it, or to drive prompting dialogues for users unfamiliar with a document type. For example, a document entry application could read the description of a figure element and invoke procedures for each element type. The procedures would issue messages to the terminal prompting the user to enter the figure ID, the depth of the artwork, and the text of the caption. The procedures would also enter the markup itself into the document being created.

Even in systems not sophisticated enough to read a type description directly, knowledge of it can be built into the application procedures. For example, in the case of the figure elements above, the figure caption procedure would terminate the figure body processing and set a switch. If the end-of-figure procedure found the switch not set it would recognize that this instance of a figure had no caption and would invoke the end-of-body procedure as well. The result is as if the markup parser had read the structure declaration, although at the penalty of increasing the complexity of the application procedures.

A.4 Conclusion

Regardless of the degree of accuracy and flexibility in document description that generalized markup makes possible, the concern of the user who prepares documents for publication is still this: can the Standard Generalized Markup Language, or any descriptive markup scheme, achieve typographic results comparable to procedural markup? A recent publication by Prentice-Hall International (6) represents empirical corroboration of the generalized markup hypotheses in the context of this demanding practical question.

It is a textbook on software development containing hundreds of formulas in a symbolic notation devised by the author. Despite the typographic complexity of the material (many lines, for example, had a dozen or more font changes), no procedural markup was needed anywhere in the text of the book. It was marked up using a language that adhered to the principles of generalized markup but was less flexible and complete than the Standard Generalized Markup Language (4).

The available procedures supported only computer output devices, which were adequate for the book's preliminary versions that were used as class notes. No consideration was given to typesetting until the book was accepted for publication, at which point its author balked at the time and effort required to re-keyboard and proofread some 350 complex pages. He began searching for an alternative at the same time the author of this paper sought an experimental subject to validate the applicability of generalized markup to commercial publishing.

---

7 The Standard Generalized Markup Language actually allows the markup to be reduced even further than this.

8 Some complete, practical document type descriptions may be found in (4), although they are not coded in the Standard Generalized Markup Language.

In due course both searches were successful, and an unusual project was begun. As the author's processor did not support photocomposers directly, procedures were written that created a source file with procedural markup for a separate typographic composition program. Formatting specifications were provided by the publisher, and no concessions were needed to accommodate the use of generalized markup, despite the marked up document having existed before the specifications./9/

The experiment was completed on time, and the publisher considers it a complete success (7)./10/ The procedures, with some modification to the formatting style, have found additional use in the production of a variety of in-house publications.

Generalized markup, then, has both practical and academic benefits. In the publishing environment, it reduces the cost of markup, cuts lead times in book production, and offers maximum flexibility from the text data base. At the same time, its rigorous descriptive markup makes text more accessible for computer analysis. While procedural markup (or no markup at all) leaves a document as a character string that has no form other than that which can be deduced from analysis of the document's meaning, generalized markup reduces a document to a regular expression in a known grammar. This permits established techniques of computational linguistics and compiler design to be applied to natural language processing and other document processing applications.

A.5 Acknowledgments

The author is indebted to E. J. Mosher, R. A. Lorie, T. I. Peterson, and A. J. Symonds—his colleagues during the early development of generalized markup—for their many contributions to the ideas presented in this paper, to N. R. Eisenberg for his collaboration in the design and development of the procedures used to validate the applicability of generalized markup to commercial publishing, and to C. B. Jones and Ron Decent for risking their favorite book on some new ideas.

A.6 References

1 B. K. Reid, "The Scribe Document Specification Language and its Compiler," *Proceedings of the International Conference on Research and Trends in Document Preparation Systems*, 59-62 (1981).
2 Donald E. Knuth, *TAU EPSILON CHI, a system for technical text*, American Mathematical Society, Providence, 1979.
3 C. F. Goldfarb, E. J. Mosher, and T. I. Peterson, "An Online System for Integrated Text Processing," *Proceedings of the American Society for Information Science*, 7, 147-150 (1970).
4 Charles F. Goldfarb, *Document Composition Facility Generalized Markup Language: Concepts and Design Guide*, Form No. SH20-9188-0, IBM Corporation, White Plains, 1980.
5 Charles Lightfoot, *Generic Textual Element Identification—A Primer*, Graphic Communications Computer Association, Arlington, 1979.
6 C. B. Jones, *Software Development: A Rigorous Approach*, Prentice-Hall International, London, 1980.
7 Ron Decent, *personal communication to the author* (September 7, 1979).

---

[9] On the contrary, the publisher took advantage of generalized markup by changing some of the specifications after he saw the page proofs.

[10] This despite some geographical complications: the publisher was in London, the book's author in Brussels, and this paper's author in California. Almost all communication was done via an international computer network, and the project was nearly completed before all the participants met for the first time.

Annex B. Implementation Considerations (Not Part of the Standard)

An implementation of the Standard Generalized Markup Language presents two faces—one to the marked up document and one to the procedures that process marked up documents.

The requirements and permitted variations for the document interface are specified in the body of this Standard. The interface to the procedures will be specified in the planned Part 7 of the Text Interchange and Processing multipart standard, entitled "Markup Support Functions." Pending the availability of Part 7, this Annex identifies markup support functions recommended for general-purpose implementations of the Standard Generalized Markup Language.

B.1 A Model of the Markup Interpretation Process

As a background for the discussion of functions, it is necessary to have in mind a model of the markup interpretation process.

> **Note:** This section should not be construed as a requirement for a particular markup parser architecture or implementation techniques. Any terms that might suggest the contrary (e.g., procedure, stack, service routine) are used only as a means of expressing functional capabilities.

B.1.1 Physical Input

The markup parser does not read a physical input stream. The parser is invoked by the system and given bit combinations to parse. When it recognizes an entity reference it tells the system, which treats the new entity as the physical input source until the entity ends, at which point the system continues processing input from the original entity.

Although the parser initiates a change of entity, it does not consider entity boundaries to be significant in the input stream (except for the requirement that a marked section must start and end in the same entity). It sees only a continuous string of bit combinations.

If the system chooses to reflect record boundary conditions to the parser, they will be handled in accordance with the Standard. In addition, if the system supports the record boundary recognition feature, the user can refer to such record boundary conditions as "pseudocharacters"; that is, as if they were bit combinations that were outside the range of the system's code (e.g., values greater than 255 in an 8-bit code). The pseudocharacters occur in the input stream in the same way as normal bit combinations.

As bit combinations beyond the range of a code cannot exist in practice, an implementation must use other techniques for representing boundary conditions, for example:

1. Representing them in the input stream as unique sequences of bit combinations, i.e., that incorporate a bit combination from the ***SYSCHAR*** category.
2. Emulating their occurrence by having the markup parser test a switch whenever it gets input from the system.

B.1.2 Scanning Modes

A significant feature of the markup language is the distinction between scanning modes. Initially, the source document is scanned in markup mode until one of the following occurrences causes a change of state:

- Descriptive markup (start-tag or end-tag):

  The GI is identified and the attributes and their values are saved. Attribute procedures mapped for immediate processing are executed. The procedure associated with the GI is then given control. It causes execution of procedures associated with the remaining attributes in any order it requires.

  Depending upon the application, an attribute or element procedure might need to read in ("absorb") all of its content before processing it, or it could simply modify the application state and then return control. When an element's procedure absorbs the content, the procedures for elements nested within it must also process their content and return it to the "absorbed" text.

- Processing instruction:

  The start is identified; the instruction and the current end delimiter (***pic***) are passed to the system. The system scans the instruction to the end, executes it, and returns control, pointing to the character after the ***pic***.

- Markup declaration or marked section:

  The start and name are identified; scanning proceeds in "declaration mode" to identify the individual parameters and the end of the declaration. The individual declaration's semantic routine is then called to execute the declaration, and control is returned.

- Entity reference:

  The end is identified; its location and the full entity reference are passed to a processing system storage manager that updates the pointer to the input buffers and returns control.

- Content:

  Text characters and unparsable data are processed as defined in "Content" on page 54. Markup mode resumes at the end of the content, or when a delimiter is encountered within it.

B.1.3 Markup Minimization

Markup minimization allows a user to omit all or part of an element's start-tag or end-tag. It requires an implementation to "track" the current location in the document structure, and, for some types of minimization, to "know" the structure in advance.

- Tracking

  The markup parser or the procedures must maintain a record of GIs of "open" explicit elements. Procedures can do this with general purpose service routines that they invoke for the elements to be tracked.

- Knowledge

  Either the markup parser must process the document type definition, or the procedures must be written to know the structure and keep track of it in addition to performing the formatting or other processing. When done by the procedures, structural knowledge requires special-purpose processing, as each element's procedure must take into account the specific elements that could precede or follow.

When the parser does not access the document type definition, the distinction between tracking and full knowledge becomes significant, as the latter puts a much greater burden on the procedures (and procedure writer!).

The possible types of markup minimization and the level of support needed for them (i.e., tracking or full knowledge) can be summarized as follows:

1. Start-tag

    a. Tag omitted, implied by:
        1) O: Preceding start-tag (KNOWLEDGE)
        2) O: Preceding element end (KNOWLEDGE)
        3) D: Tag implied by text (KNOWLEDGE)
    b. N: Null-tag (TRACKING)

2. End-tag

    a. Tag omitted, implied by:
        1) O: Following start-tag (KNOWLEDGE)
        2) N: Following null start-tag (TRACKING)
        3) E: Containing element end-tag (TRACKING)
        4) D; Text (KNOWLEDGE)
    b. N: Null-tag (TRACKING)

Note that implying the ending markup from the end of a containing element requires that both the current element and the containing element be tracked. The other cases supported by tracking require that only the current element be tracked.

B.1.4 Translation

The Standard assumes that the document's character set is the same as the system's, i.e., that the markup parser need not be cognizant of any translation. Any required translation should be done by the system prior to processing, or during processing in a manner that is transparent to the markup parser.

B.2 Markup Support Functions

The processing system should provide initialization support and make a number of functions available for use by procedures.

B.2.1 Initialization

An implementation must make certain information available to the markup parser at the start of a run. The manner of passing the information is system-specific, but the information must include the following:

- The mappings between GIs and attribute names, on the one hand, and the procedures to be executed for them, on the other.

  This could be in the form of the name of a single procedure (e.g., the procedure for the document level GI) which would set up the mappings for the other elements.

- The coexistent structures, if any, to be processed.

  If more than one structure is to be processed, additional mappings of GIs and attribute names will be required for the additional structures.

B.2.2 Structure Processing

It is recommended that the markup parser manage tracking of the structure and processing of the **STRUC** and **ATT** declarations to determine whether the markup conforms to the document type definition. This arrangement will allow markup minimization without placing a large burden on the procedure writer. At a minimum, the implementation should provide system functions to make it easier for the procedures to perform these tasks themselves.

B.2.3 Associating GIs and Attributes with Procedures

An implementation must provide functions for associating (mapping) GIs and attribute names with particular procedures.

If structural processing is not done by the markup parser, it must be possible to specify, as part of the mapping, information that would normally be available from the document type definition, such as whether attributes and/or minimization are allowed.

Mapping must be supported as a dynamic process, available to procedures, since, in some applications, the procedure for a given generic identifier (i.e., element type) might include the specification of processing for the elements of its content. For example, in a formatting application, the procedure for an element of the type "body" might specify that elements of the type "heading" are to be set in a Roman font, while the procedure for the generic identifier "appendix" might specify that "heading" elements occurring within it should be set in italics.

In other words, the heading element can be said to be "qualified" (as a COBOL structure is qualified) by the type of elements within whose bounds it occurs. The model of a surname and personal name suggests itself: the heading shares something in common with all other headings (same personal name), but that is modified by the fact that it is part of a body or an appendix (two different surnames).

When mapping attribute procedures, it must be possible to specify whether they are to be executed by the parser before the GIs procedure or under the latter's control.

B.2.4 Functions Available to Procedures

A general-purpose text processing or word processing system should make the following capabilities available to procedures:

1. Procedure creation, with logic and branching capabilities.
2. Attribute processing, including interrogating the values of specified attributes and executing the attribute procedures.
3. Processing state interrogation, including:
    a. identifier and physical type of the current entity (file, macro, etc.);
    b. current delimiter role assignments, etc.;

   c. current location in the document structure (if tracked by the system) so that, for example, the procedure for a heading could tell whether it was in the front matter of a book or the body.

4. Storage and retrieval of attribute values and elements (content and/or tags), with or without markup parsing.

5. Insertion of literal text or entities into the source document (with or without markup parsing) or into the output. This includes putting markup and declarations into the source document.

6. Direct execution of markup declarations.

7. Extended definition of markup entities, such as setting them with an associated font in which the entity will appear when printed. These functions are for use in procedures and processing instructions, not as extensions to the **ENTITY** declaration syntax.

8. Preprocessing of document type definitions to improve performance (instead of interpreting the **STRUC** and **ATT** declarations dynamically each time a document is processed).

Annex C. Markup Construct Registration (Not Part of the Standard)

The descriptive markup portion of the Standard Generalized Markup Language is a metalanguage. That is, although it unambiguously identifies such things as generic identifiers, attributes, and entity references, no *specific* GIs or other names are part of the language. The vocabulary is made up by users to meet their needs, and is defined in the structure and attribute declarations.

Maximum benefit will be derived from the Standard if trade organizations, technical societies, and similar groups with an interest in the interchange of specialized material have some method of sharing information about document types and other aspects of the language. To this end, the planned Part 10 of the Text Interchange and Processing multipart standard will specify registration procedures for text processing constructs, including markup constructs. Interested organizations could use the standard procedures, or their own, or even accomplish many of the benefits of registration by simple agreement among users who regularly exchange documents.

This section describes some of the markup constructs that could beneficially be the subject of registration procedures or interchange agreements.

C.1 Document Type Definitions

A document type definition is a formal specification of the valid markup of a class of documents that shares a common generic identifier. Part of the definition is expressed in structure, attribute, and entity markup declarations; the remainder is narrative for human consumption (for example, the kind of data content notations used).

The entity declarations refer to entities with a constant value that are used in that type of document; for example, unkeyable graphic symbols.

Document type definitions (or, more properly, element type definitions) can also be created for elements that normally do not exist as independent documents, but which can occur in a variety of document types.

C.1.1 Common Document Types

The following document types are under consideration for initial registration:

1. ISO Standard

2. ANSI Standard

3. Technical report (ANSI/ISO working papers)

4. ANSI X3 Standing Document for project reports (SD4).

C.1.2 Common Element Types

There are certain elements that have well-recognized common forms that can occur in a variety of documents. Some involve specialized data content notations as well (see below).

Some possible candidates for registration and sources of existing specifications are:

1. Legal citations (Harvard Law Review)

2. Mathematical formulas (American Mathematical Society)

3. Chemical formulas (American Chemical Society)

4. Bibliography entries (Library of Congress)

5. Name and address (direct mail organizations, directory publishers, telephone companies)

C.1.3 Pro Forma Document and Element Types

There are a number of element configurations that occur in a variety of documents and elements, but usually with differences in the GIs, or other variations. *Pro forma* descriptions of such configurations could be registered so they would be available to serve as guides for constructing specific descriptions that a user might require.

Examples of such elements are:

1. Paragraph (for typical formatting applications)
2. Paragraph (for detailed syntactic or other analysis)
3. Tables
4. Nested ranked paragraph structures
5. Lists
6. Overall document structures

C.2 Data Content Notations

In the Standard Generalized Markup Language, data content is the part of a document that is processed by the application procedures. Although each procedure could conceivably use a different data content notation, in practice there are commonly used ones that could beneficially be adapted for use with the Standard Generalized Markup Language and registered.

For example, the popular EQN mathematical notation could be adapted as follows:

1. The processing commands that indicate the start and end of an EQN statement are not needed, as their function is handled by generic identifiers and attributes.

2. The processing commands for defining new EQN words are not needed because that function could be handled by entity references.

3. The remainder of EQN, which is the actual data notation, could be used as is, or possibly supplemented by explicit markup for some of the mathematical elements.

```
<p>The formula,
<q><formula notation=EQN>E equals m
times c squared</formula></q>
should provide speedy enlightenment.</p>
```

C.3 Entity Sets

It has been estimated that there are over 20,000 graphic symbols used in the publishing of text. It would be useful to register sets of accepted entity names for these symbols, organized according to subject matter (e.g., chemical, mathematical, etc.). Document type designers could then use the registered sets instead of creating new names for the same symbols, thereby providing a uniform entry convention regardless of the keyboard or character set used.

C.4 Word Processing Conventions

Word processors attempt to offer text entry operators an interface similar to that of the typewriters on which many operators were trained. On typewriters, each key produces an immediate formatting result as it is struck: a carriage return starts a new line, a tab key generates horizontal space, and so on. Hence typewriters, and the word processors patterned after them, are sometimes known as "WYSIWYG" devices—for "what you see is what you get."

Such keystrokes are specific processing instructions. As such, they limit the use of the document to a single formatting style, executable only on the machines that understand the instructions.

The Standard Generalized Markup Language, though, can offer the benefit of familiar typewriter keyboarding while still maintaining the generality of the document. When the self-delimiting entity reference feature is supported, typewriter keystrokes can be interpreted as descriptive markup. For example, the declarations

```
<!ENTITY  H/09   --TAB    -- "<p>"  >
<!ENTITY  H/0D0A --CR, LF-- "<p>"  >
```

map both the tab and the carriage return, line-feed sequence into the start-tag for a paragraph element. The actual formatting for a paragraph depends, as always, on the application procedure, just as if the starting markup had been entered explicitly.

The presence of the entity declarations, in effect, turns typewriter keystrokes into generalized markup. The entry device, though, can still produce the tabbing or new line called for by the keystrokes, thereby giving the user the immediate visual feedback that is such an important aspect of these systems.

Moreover, for complex elements, such as multipage tables with running headings, where typewriters do not offer a convenient and generally accepted entry convention, the typewriter keystrokes can be supplemented with explicit generalized markup.

To facilitate interchange with word processors, it would be useful to register sets of entry conventions, in the form of Standard Generalized Markup Language entity declarations.

Annex D. Nonconforming Variations (Not Part of the Standard)

Historically, many of the benefits of generalized markup have been obtained by using a procedural text processing language as though it were a descriptive language. That approach is necessarily less effective than using the Standard Generalized Markup Language, which was designed specifically for the purpose of describing documents.

This Annex discusses some common departures from the Standard that occur when procedural languages are used for descriptive markup.

D.1 Fixed-length Generic Identifiers

In a fixed-length GI language, no attributes are supported, and there is no ***esmc*** or ***eemc***. The *content* begins with the first character after the start-tag GI.

The following example uses 2-character GIs:

```
<paThis is a paragraph with
a <sqshort quotation</sqin it.</pa
```

Some drawbacks to fixed-length GIs are:

- The source document can be difficult for a human to read (although machines have no trouble).
- Some "natural" tag mnemonics will be unusable because of their length (e.g., "p" for paragraph and "h1" for head-level-1 cannot both be used).
- Document types that have attributes cannot be processed.

A document created with fixed-length GIs can be converted to a Standard Generalized Markup Language document (provided no other differences exist) by inserting the ***esmc*** and ***eemc*** delimiters.

D.2 Single Delimiter

In a single delimiter language, the same character is used for the ***esmo***, ***eemo***, ***ero***, ***mdo***, and ***pio*** delimiter roles.

This approach introduces a number of restrictions:

- GIs, entity names, declaration names, processing instruction names, and processing macro names must all be different from one another, as there is no other way to distinguish them.
- End-tags can be entered only when the context offers no possibility of confusing them with start-tags, as they look the same.

Some drawbacks to the use of a single delimiter are:

- The structure of the document is no longer apparent to a human reader because descriptive tags cannot easily be distinguished from processing instructions, markup declarations, or entity references.

- Processing instructions, which apply only to a single application and system, cannot easily be located for modification if the document is used in a different application or system.

- A document type designer must be aware of every processing instruction and macro name and cannot use the same names for GIs. Similarly, a text programmer cannot create a new macro without making sure that no GI exists with the same name.

- Document interchange is severely limited because of the difficulty in preventing a document's GIs from conflicting with a receiving system's macro names.

An obvious solution to these problems is to use naming conventions to distinguish the types of name. For example, declaration names could begin with an exclamation point, processing instruction names with a question mark, and so on.

```
<!MS IGNORE>
<?STORE date as SYSDATE>
```

The Standard Generalized Markup Language achieves the same result as a single delimiter with naming conventions by allowing multiple-character strings as delimiters.

ANNEX Z

Information Processing Systems - Programming Languages - Text Interchange and Processing - Part One: General

Sixth Working Draft, International Standard

ISO TC97/SC5/EG CLPT - X3J6 N177-1 1983 Jun 01

[BLANK PAGE]

ANNEX Z

Information Processing Systems—
Programming Languages—
Interchange and Processing of Text

PURPOSE

The purpose of this section of this document is to provide the domestic and international printing and publishing industries with access to parts of the standard not directly related to Part VI, and to place all parts in contextual reference.

REFERENCES

ISO working draft documents comprising the entire standard as submitted include:

TC97/SC5/EG CLPT—X3J6 Part One: General	N177-1 Sixth Working Draft	*See Note*
TC97/SC5/EG CLPT—X3J6 Part Two: Vocabulary	N177-2 Sixth Working Draft	*See Note*
TC97/SC5/EG CLPT—X3J6 Part Three: Programming Language	N177-3 Sixth Working Draft	
TC97/SC5/EG CLPT—X3J6 Part Four: Editing	N177-4 Sixth Working Draft	
TC97/SC5/EG CLPT—X3J6 Part Five: Formatting and Composition Factors	N177-5 Sixth Working Draft	*See Note*
TC97/SC5/EG CLPT—X3J6 Part Six: Document Markup Metalanguage	N177-6 Sixth Working Draft	*See Note*

NOTE: Proposed as ISO Draft International Standard

DEVELOPMENT

This standard was developed and continues to be developed under the regular policies, organization, and procedures of:

1. **American National Standards Committee X3J6**

   X3 — Information Processing Systems
   J6 — Computer Languages for the Processing of Text

   Project 203

and 2. **International Organization for Standardization (ISO)**

   TC97/SC5/EG CLPT
   TC97 — Information Processing
   SC5 — Programming Languages
   EG — Experts Group
   CLPT — Computer Languages for the Processing of Text

   Project 97.5.12

[BLANK PAGE]

NOTE

The provisions included within this document may change as a result of the review and coordination process within the International Organization for Standardization (ISO) and the American National Standards Committee X3, Information Processing Systems.

SPECIAL NOTE

Parts One and Six

Provided in full in this document.

Parts Two through Five

Full-text copies may be obtained from:

Ms. Catherine A. Kachurik
Director, X3 Secretariat
Computer & Business Equipment Manufacturers Association (CBEMA)
311 First Street, N.W., Suite 500
Washington, D.C. 20001
202/737-8888

Parts Seven through Ten

These parts of the Standard are still under development.

[BLANK PAGE]

Information Processing Systems – Programming Languages – Text Interchange and Processing – Part One – General

Expository Remarks

(These expository remarks are intended to be removed after the Technical Committee deliberations are completed, and are thus not a part of the standard)

The standard was drawn over a period of five years. The developers came from a broad spectrum of backgrounds. Relatively independent elements evolved which are presented in several parts of the standard of which this is part one.

Part one (General) is the connective tissue of the standard. It declares the relationships of the parts.

Part two (Vocabulary) offers two ingredients. The first ingredient is the vocabulary of the information processing personnel dealing with text processing applications. The second ingredient is a vocabulary of text processing terminology for use by information processing specialists who might not be familiar with the field. Part two, in its entirety, is also intended as input for the Information Processing Systems Committee on Vocabulary (ISO/TC 97/SC 1).

Part three (Programming Language) forms the basis for an application which deals with text. It is a language intended for both beginners and specialists. It was designed for implementation as an interpretive processor. A compiler implementation is not precluded, though some features may not be as feasible in that situation. The information in parts four to nine is intended to be expressible in the language of part three and, indeed, parts seven and nine are specific examples of the use of part three in practice. The language draws from several sources. It is intended as a text manipulator's BASIC with many of the features of a Pascal or Ada without the restrictions they impose.

Part four (Entry and Editing Functions) is a description of an application of editing in terms of, but not demanding implementation in the Programming Language.

Part five (Formatting and Composition Functions) is a description of an application of formatting in terms of, but not demanding implementation in the Programming Language.

Part six (Document Markup Metalanguage) is independent of part three. Part six defines a means of describing the structure of a document while avoiding any reference to the processing that the structurally marked document might undergo. This offers two very important advantages. First, the author is relieved of concentrating upon the appearance of the document and can concentrate on its structure and content. Second, linkages can be developed to permit altering the environment in which a document is processed from a word processor to a publication system without changing the document itself.

Part seven (Markup Support Functions) provides the application of part six in the language of part three. By separating the binding from the fundamental markup structure, advantage may be taken of the concepts using existing text processors which were never intended to be compatible with the Programming Language. With the binding, the features of the parts may be combined into a single application.

Part eight (Binding to the Graphical Kernel System) provides the capability of containing within one data set the entire abstract version of a document inclusive of pictures. This is a far cry from either facsimile or traditional paste-up. For those authors capable of thinking in terms of their entire document, and for those publishers who wish to control editions of documents down to the last dot, part eight coupled with the other parts, offers a solution long sought.

Part nine (Application to "What You See is What You Get" (WYSIWYG) Processing) covers the linkage of the concepts of the word processor and the remainder of the text processing community. Contemporary word processors employ techniques which can be mapped upon the Programming Language and the Document Markup Metalanguage. It is expected that such mappings may be used to migrate to newer systems and permit users to establish their own operating standards without being coerced by the features of certain products. An important aspect of this effort is the establishment of examples of how the tricky features of existing products can be performed in environments other than their original hosts. The human factors aspects of the WYSIWYG process lead to the prediction that the part nine will be a bridge to future extensions.

Part ten (Registration Procedures) offers a mechanism for interchanging the description of parts of document design without being forced to be restrictive about any of the stylistic aspects of documents. The registry of elements would not standardize elements, but where a user found an element that was registered he would be relieved of the effort of designing a duplicate.

Given these parts, the preparation of documents as well as many other office and business activities can be simplified. By avoiding the specific physical linkages in developing the standard, the technological trap of infeasibility is avoided. The standard describes the processes in use which are expected to survive the onslaught of electronic technology. It bridges from today's office into an office of the future where the reference document standard has shifted from today's typewritten page to a multicolor letterpress quality image.

[BLANK PAGE]

Foreword

This foreword is not a part of this International Standard.

This standard is entitled "Information Processing Systems - Programming Languages - Text Interchange and Processing." The first portion of the title represents the title of the parent body TC 97 and correctly indicates the broad field of application. The second portion identifies the area of development and Sub-Committee of TC 97 to which the development was assigned, namely SC 5. The remaining portion of the title was derived from the project title with the words "Computer Languages" deleted because of redundancy with the previous words.

The project for the standard Computer Languages for the Interchange and Processing of Text was initiated by TC 97 in 1979 and assigned to An Expert Group by SC 5 - Programming Languages. The charter was very broad and included material which is outside the usual scope of a programming language. During the first three years of development a single document was envisaged. As the publication approached it became obvious that the using audience would be better served by a single standard developed and approved in parts. Therefore, this standard has been prepared in parts.

No defacto standards existed at the start of the project and none had appeared by the time of the approval process. In fact, a marked increase of the publication of differing packages with similar characteristics started in 1981 and continued. The standard is drawn to be compatible with the Open System Interconnection Model. It functions as a member of the Application Layer of that Model.

The presentation requirements of a programming language are not always served best by the style guides of a publication. So it is with this standard. Where necessary to assist clarity, the usual two column format of an ISO Standard is not used. Instead, those clauses which are better presented in a wider format are presented in single column format. The several type styles used in those clauses also would create a reader confusion if not compatibly carried throughout the standard. Thus, a larger type font with serifs is used throughout the standard.

[BLANK PAGE]

1 Introduction

The project "Computer Languages for the Interchange and Processing of Text" was initiated to develop a standard covering a very broad charter. This is part one of a multiple part standard. All of the parts are related to this part, though some of the later parts are not directly related to each other.

Material was provided to the development project team from several sources. Key sources are listed below:

— RUNOFF; *Dr. J. Saltzer*; Massachusetts Institute of Technology
— NROFF; Bell Laboratories
— Tau Epsilon Chi; *Dr. Donald Knuth*; Stanford University
— TECO; Digital Equipment Corporation
— EMACS; Massachusetts Institute of Technology
— EDT; Sperry Corporation
— TEXTFORM; University of Alberta
— PTP; Summer Institute of Linguistics
— MINCE; Mark of the Unicorn
— GenCode; Graphic Communications Association
— COMPOSE; *Wallman*; Honeywell Inc.
— TEX; *Bemer, Clamons, & Keyes*; Honeywell Inc.
— EDIT; University of Michigan
— NICOLA; University of Dortmund
— Ada; United States Department of Defense
— SCRIBE; Carnegie Mellon University
— GML; International Business Machines
— NLS; Stanford Research Institute
— COSCL/JOD; CODASYL
— Software Tools; *Kernighan and Plauger*; Addison Wesley

Text processing ranges in application from Word Processing to Message Networking to Composition systems. The effort in developing this multiple part standard was to treat each diverse area of the field such that interchange between applications is enhanced.

It is intended that the standard be implemented in many environments. It is also intended that the environments must adapt to the standard. The standard requirements were not reduced to meet the common denominator of services offered in contemporary systems. The standard does not address the input device (such as a keyboard). It also does not address the output or presentation device(s) or media. While it addresses the coded representation of the text and the language, this is done to solidify the abstraction and ensure proper portability.

2 Scope

This standard is in multiple parts that collectively define languages and functions for the definition, description, entry, recording, recall, transformation, searching, manipulation and display of text, with compatible overlap in the areas of word processing and computer graphics.

Each part of the standard contains a clause that defines its own scope.

3 Field of Application

In general, the standard can be implemented in any environment with sufficient functionality, including any combination of:

— A free standing device dedicated to text processing or word processing.
— An interactive device that, in combination with a host, emulates the effect of a free standing device.
— A batch processing application in a data processing system.
— A transaction based system.
— A personal computer.

There is no dependency on specific display devices, data entry devices, data streams, or physical file organizations.

The standard applies in the Applications layer of the Open System Interconnection Reference Model.

Each part of the standard contains a clause that defines its own field of application.

4 References

Each part of the standard contains a clause that lists other standards, including parts of this standard, on which it is dependent.

5 Organization

The standard contains the following parts:

1 General
This is the part you are reading. It describes the overall scope and field of application of the standard, and the subjects of the individual parts.

2 Vocabulary
This part contains a general vocabulary of text processing terms, as well as the specialized vocabulary used in other parts of the standard.

3 Programming Language
This part specifies a text processing programming language for applications that deal with text. It can be used to implement functions defined in other parts of the standard, although that is not a requirement. The language can be implemented as an interpreter or as a compiler, and is intended for both end users and programmers.

4 Entry and Editing Functions
This part specifies functions commonly used in text entry and editing applications. The specification is in terms of the text processing programming language, but the standard does not require that the functions be implemented in that language.

5 Formatting and Composition Functions
This part specifies functions commonly used in text formatting and composition applications. The specification is in terms of the text processing programming language, but the standard does not require that the functions be implemented in that language.

6 Document Markup Metalanguage
This part specifies a notation that describes the structure and attributes of documents generically, without respect to the processing that they will undergo. The documents can therefore be used with a multiplicity of systems, devices and applications.

7 Markup Support Functions
This part specifies functions needed by applications that process documents prepared with the markup language. The specification is in terms of the the text processing programming language, but the standard does not require that the functions be implemented in that language.

8 Binding to the Graphical Kernel System
This part specifies the binding of the Graphical Kernel System to the text processing programming language, thereby allowing the integrated processing of illustrations and text.

9 Application to "What You See is What You Get" (WYSIWYG) Processing
This part specifies how word processing interfaces and functions can be implemented in the text processing programming language.

10 Registration Procedures
This part specifies registration procedures for text processing constructs defined in other parts of the standard. Registration minimizes duplicate development, and facilitates shared use, of constructs that are inappropriate to standardize.

A Acknowledgements

This annex is not a part of the standard.

The participation of the following individuals in the development process is acknowledged. They were not active participants at the start of the approval cycle, but made significant contributions in critical early stages.

Robert W. Bemer
Louis O'Korn
Walter Lapinski
Sharon Crook
Ke Shih
Charles Lightfoot
Norman J. Hunt
Diane L. Smolenski
Shirley W. Watkins
Donald Knuth
Sandy Cain
Judith Schlessinger
Micheal Lederer
Blanton C. Duncan
Gerald M. Wattenberger
Sharon Murry
George Oerter
Alan R. Wylie

The following were the individual experts who participated in the development of this standard:

> Note: A tabulation of names organised by country represented will be placed here when the DIS stage is reached.

The inclusion of such acknowledgement is unusual in an ISO standard. Similarly, the development of a new computer language is also unusual under the ISO aegis. While the major participation is visibly from North America, all contributions have been considered. As an example, the PTP contribution was developed in Papua, New Guinea.

GCA

STANDARD 101-1983

Document Markup Metalanguage (S G M L)

Adopted by
DEPARTMENT OF DEFENSE
10 August 1983

END

Graphic Communications Association
1730 North Lynn Street / Suite 604 • Arlington, Virginia 22209 • Phone (703) 841-8160

An Affiliate of Printing Industries of America, Inc. PIA INC

GCA

GCA

GENCODE™

And The

STANDARD GENERALIZED MARKUP LANGUAGE

(S G M L)

GCA

Graphic Communications Association
1730 North Lynn Street / Suite 604 • Arlington, Virginia 22209 • Phone (703) 841-8160

An Affiliate of Printing Industries of America, Inc.

[BLANK PAGE]

Foreword

Part B presents expository material primarily intended to be a conceptual tutorial. The Model Contract Specification illustrates existing utilization practices—the planning, thought, and production techniques of which contributed to the usefulness of the provisions in the standard. The GenCode™ Primer has been, and remains, a good "first-exposure" illustration. The individual articles, in toto, all reflect actual applications experience. Kit von Sück and James Mason utilize the standard itself as the basis for discussion and practical example. The remainder reflect experience with earlier forms of generic coding. All are based upon thought, planning, experiment, and useful application. They do not in all respects represent compliance with the provisions of the standard, but they give practical illustration of the concepts.

These articles individually and collectively illustrate the importance of the clear separation of form and content—to yield greatest flexibility and maximum utilization. They all reflect the importance of editorial discipline—a factor the reader is encouraged to bear in mind constantly. Experience with typographic composition—output-form-driven (display-oriented)—makes it all too easy to take convenient "escapes" from editorial discipline. For example, there is the constant temptation to code for "italics" (visual form) (because it is seemingly easier and quicker) rather than "legal citation", "emphasis", or "Latin phrase" (editorial content). There is a logical structure to typographic coding—there are even generic typesetting codes. If the reader will but run his global search for "italics" on the one hand, and his global search for "legal citations" on the other, he will rapidly conclude that the *LOGICAL STRUCTURE OF INTEREST IS THE* ***EDITORIAL STRUCTURE!***

Upon initial consideration, the reader might conclude that these seemingly small distinctions—and the stress on editorial discipline—are differences of style rather than substance. With thought, and trial—particularly with multiple output forms: e.g., visual-image output, line-printer output, CRT output, data-base output—he will conclude that the distinctions are substantial in their processing impact. The more that one views "information production" as the basis process, the more one will be convinced of the power of editorial logic, editorial discipline—a content-driven world as opposed to a form-driven world.

WILLIAM W. TUNNICLIFFE
Vice-President
Information Technologies
Graphic Communications Association

[BLANK PAGE]

GCA

GENCODE™

MODEL CONTRACT SPECIFICATION

GCA

Graphic Communications Association
1730 North Lynn Street / Suite 604 • Arlington, Virginia 22209 • Phone (703) 841-8160

An Affiliate of Printing Industries of America, Inc.

[BLANK PAGE]

Model Contract Specification

Internal Revenue Service Specifications for Employment of Generic Coding— Three Levels of Implementation

William W. Davis

Computer Specialist/Procurement
Publishing Services Branch
Internal Revenue Service

and

Member
ANSI X3J6

Chairman
ANSI Task Group X3J6.1
EG CLPT Task Group SGML

Material prepared for IRS use
1982 November 24

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the GCA copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Graphic Communications Association. To copy otherwise, or to republish, requires a fee and specific permission. © 1983 GCA

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

Editor's Note

The material which follows is an excerpt from the IRS Solicitation, Offer and Award IRS-P-83-5, issued 1982 November 24.

Material which bears directly on GenCode™ concepts and implementation has been included in the excerpt: Face sheet for exact reference plus pages 10 through 67 of 74.

Material not considered to be as directly related has been omitted. Material omitted includes:

	Standard Form No.	Pages of Form	Pages of 74
Before Page 10 of 74:			
	33	2, 3, 4	2, 3, 4.
	33-A	1, 2 plus 9 & 13 detail	5, 6, 7.
	36	8, 9	8, 9
After Page 67 of 74:			
	32	1, 2, 3, 4, 5	68-72
Supplemental Clauses Fixed-Price Supply Contracts			73, 74

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

SOLICITATION, OFFER AND AWARD

3. CERTIFIED FOR NATIONAL DEFENSE UNDER DPS REG. 1 AND/OR DMS REG. 1 RATING	4. PAGE 1	OF 74	
1. CONTRACT (*Proc. Inst. Ident.*) NO.	2. SOLICITATION NO. IRS-P-83-5 ☒ ADVERTISED (IFB) ☐ NEGOTIATED (RFP)	5. DATE ISSUED 11-24-82	6. REQUISITION/PURCHASE REQUEST NO. To be assigned with Delivery Orders
7. ISSUED BY CODE Department of the Treasury - Internal Revenue Service Washington, DC 20224		8. ADDRESS OFFER TO (*If other than block 7*) Internal Revenue Service - PM:S:FM:P Publishing Services Branch, Room 1553 1111 Constitution Ave., N.W. Washington, DC 20224	

In advertised procurement "offer" and "offeror" shall be construed to mean "bid" and "bidder"

SOLICITATION

9. Sealed offers in original and 3 signed copies for furnishing the supplies or services in the Schedule will be received at the place specified in block 8, *or if handcarried, in the depository located in* Room 1553, IRS Building until 11:00 am (*Hour*) *local time* 12-10-82 (*Date*)

If this is an advertised solicitation, offers will be publicly opened at that time.

CAUTION — LATE OFFERS: See pars. 7 and 8 of Solicitation Instructions and Conditions.

All offers are subject to the following:

1. The Solicitation Instructions and Conditions, SF 33-A, Rev. 1-78 edition which is attached or incorporated herein by reference.
2. The General Provisions, SF 32, 4-75 edition, which is attached or incorporated herein by reference.
3. The Schedule included herein and/or attached hereto.
4. Such other provisions, representations, certifications, and specifications as are attached or incorporated herein by reference. *(Attachments are listed in schedule.)*

FOR INFORMATION CALL (Name & telephone no.) (No collect calls) ▶ Paul Perry (202) 566-9807

SCHEDULE

10. ITEM NO.	11. SUPPLIES/SERVICES	12. QUANTITY	13. UNIT	14. UNIT PRICE	15. AMOUNT
	Electronic photo composition and related services.	Enter pri	ces	in section 1	5

See continuation of schedule on page 4

OFFER *(pages 2 and 3 must also be fully completed by offeror)* also page 4

In compliance with the above, the undersigned agrees, if this offer is accepted within ______ calendar days *(60 calendar days unless a different period is inserted by the offeror)* from the date for receipt of offers specified above, to furnish any or all items upon which prices are offered at the price set opposite each item, delivered at the designated point(s), within the time specified in the schedule.

16. DISCOUNT FOR PROMPT PAYMENT *(See par. 9, SF 33-A)*

___ % 10 CALENDAR DAYS, ___ % 20 CALENDAR DAYS, ___ % 30 CALENDAR DAYS, ___ % ___ CALENDAR DAYS

17. OFFEROR *CODE* ___ *FACILITY CODE* ___ NAME AND ADDRESS *(Street, city, county, State and ZIP code)* AREA CODE AND TELEPHONE NO. ▶ ☐ *Check if remittance address is different from above — enter such address in Schedule*	18. NAME AND TITLE OF PERSON AUTHORIZED TO SIGN OFFER *(Type or print)* 19. SIGNATURE 20. OFFER DATE

AWARD *(To be completed by Government)*

21. ACCEPTED AS TO ITEMS NUMBERED	22. AMOUNT	23. ACCOUNTING AND APPROPRIATION DATA
24. SUBMIT INVOICES *(4 copies unless otherwise specified)* TO ADDRESS SHOWN IN BLOCK ______		25. NEGOTIATED PURSUANT TO ☐ 10 U.S.C. 2304(a) ( ) ☐ 41 U.S.C. 252(c) ( )
26. ADMINISTERED BY *(If other than block 7)* CODE		27. PAYMENT WILL BE MADE BY CODE Internal Revenue Service Accounting Section, PM:S:N:F, Rm. 6611 Washington, DC 20224
28. NAME OF CONTRACTING OFFICER *(Type or print)*		**29. UNITED STATES OF AMERICA** **BY** *(Signature of contracting officer)* — 30. AWARD DATE

Award will be made on this form, or on Standard Form 26, or by other official written notice

[BLANK PAGE]

Internal Revenue Service Specifications for Employment of Generic Coding—Three Levels of Implementation

Table of Contents

Section 1—Scope 127
Section 2—Term of Contract 127
Section 3—Basis of Award 127
Section 4—Contract Administration Contract 128
Section 5—Order Procedure 128
Section 6—Schedule 129
Section 7—Pickup and Delivery of Material 129
Section 8—Security 129
Section 9—Supplies 130
Section 10—Proofreading 130
Section 11—Description of General Requirements and IRS Standards 130
Section 12—Description of Composition Requirements 133
Section 13—Description of Composition Related Requirements 135
Section 14—Explanation of Charges 136
Section 15—Schedule of Charges 138
Section 16—Quality Factors Capabilities 141
Section 17—Definitions 141
Attachment 1—Form M-5905, Electronic Composition Work Request 143
Attachment 2—IRS Standard Character Set 144
Attachment 3—Linotron Locator Log 146
Attachment 4—FQGI and Symbol Descriptions 147
Attachment 5—Format Change Notification 148
Attachment 6—IRS Printout Standards 149
Attachment 7—Category One Sample 150
Attachment 8—Category Two Sample 152
Attachment 9—Category Three Sample 153
Attachment 10—Category Four Sample 155
Attachment 11—Category Five Sample 157
Attachment 12—Key Word in Context Printout 162
Attachment 13—Table Layout Sample One 163
Attachment 14—Table Layout Sample Two 164

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

SECTION 1—SCOPE

1) This is a multiple award, open-end contract for the purchase of computerized electronic photocomposition. The finished product, including all applicable elements, will be purchased in one of two ways; either a driver tape for full-page straight-in typesetting via the Government Printing Office's VideoComp 500, or full-page reproduction proofs run on the contractor's phototypesetter. (This would apply only in specific instances where IRS has been granted a waiver by GPO.) All production will require a file tape in the IRS masterfile format except for Category Four composition (tables).

2) Because of the open-ended nature of work to be produced under the provisions of this contract, any given job requires its own unique combination of production elements. The various elements required for a specific job can include file creation, initial keyboarding, codification, file updating, extracting, sorting, global changes to file, typesetting, etc.

3) Generically coded files are provided for in this contract. This coding technique identifies textual elements and heirarchies within a document based on usage or type rather than typographic characteristics. To implement this text processing approach, two types of masterfiles are allowed in addition to the standard locator file.

4) Based on past requirements, IRS expects to purchase approximately 17,000 pages during the contract term.

SECTION 2—TERM OF CONTRACT

1) The term of the contract will be from January 3, 1983 or the date of award, whichever is later, through December 31, 1983.

2) Contractors must accept jobs on this contract until the end of the contract period, even though some portion of the work may be performed after that date.

3) The IRS is currently operating under a 'continuing resolution' funding appropriation. Under this resolution, funds are available only through December 17, 1982. Notwithstanding the IRS' intention of executing a contract through the period ending December 31, 1983, this contract does not obligate the IRS for any service charges beyond December 17, 1982. This contract will automatically become effective January 3, 1983 or on the date of award, whichever is later, and will terminate on December 31, 1983 unless the IRS notifies contractor's by December 16, 1982, that it will not become effective. Such notification of discontinuance shall not subject the IRS to any 'termination', 'cancellation', or 'discontinuance charges'.

SECTION 3—BASIS OF AWARD

3.1 Multiple Award

1) It is the intention of the IRS to certify all contractors who qualify, since it is anticipated that one contractor will not have the capacity or capability to satisfy all the requirements of this contract. Since it is not possible to determine the exact requirements and needs of the IRS at this time, contractors should be aware that there are *no* guarantees specified for the amount of work any contractor will receive.

2) In addition, no contractor will be certified as the low contractor for this contract. To determine the sequence of contractors for purposes of offering work, the cost of each individual job or logical grouping of jobs will be estimated from the prices of those contractors who bid on providing the required services (see Section 15).

3.2 Contractor Certification

1) In order to meet the needs of the IRS, the Contracting Officer will certify contractors for those categories of the contract where the contractor has placed a bid price and has demonstrated to IRS that he/she has the capability and the programming completed to perform. To this end, the contractor who has not yet completed any of the programs required to do any one or more of the requirements for this contract may place a price on the pricing sheet if that service can be made available to IRS within 90 days after the bid opening. (The 90-day requirement does not apply for the programming required for masterfiles B and C.) If the contractor has completed all systems development and has demonstrated that capability by the aforementioned deadline, the IRS will then certify the contractor and issue his bid price. All testing of the contractor's system will be at the contractor's expense, except for GPO charges.

3.3 Statement of Experience, Qualification and System Description

1) All bidders who have not done so in the past 24 months or who have had significant system changes must submit a statement of experience and qualification and a description of their systems and capabilities with the samples required in subsection 3.4.

2) To facilitate evaluation of the proposal, each offeror shall submit the information requested in the following sequence:

a) Relevant experience of proposed staff in the field of data processing, preparation of computer files for photocomposition, and in the use of GPO phototypesetters. Also, experience of job planners (especially typographic experience) and proofreaders.

b) System Description—include the following, plus any additional information, which might be required to describe adequately the contractor's capability:

   1) Copy Markup

   2) Keyboarding—include method of keying locators, generic identifiers, special characters, typeface changes, etc.

In case of conflict between any statement in any article and the standard, the standard governs.

3) Computer Printout (CPO)—include a description of how material is identified on the CPO for updating purposes. Adequate identifiers must be on the CPO to allow keyboarders to update or correct the file directly from the CPO without further lookup or transcription of corrections to another CPO or device. Attach a sample page that indicates the format of the CPO that will be used.

4) Correction Cycle—include proposed methods of proofing, identifying location of mistakes, keyboarding corrections and method of assuring that only keyboarding corrections are put into file. Include all procedures that are used to assure that no extraneous material is added to the file, or that no material is inadvertently deleted from the file.

5) Edit Programs—include a description of the programs that convert tapes in the IRS standard masterfile format to the contractor's processing format and back. Include the capabilities of the program to handle multiple formats and conversion of a file from one format to another. Also include the capability of producing driver tapes.

6) Edit programs to produce a tape for another typesetting system—if available, state the time frames and capability of the contactor's system to convert from the IRS standard masterfile tape into a tape for another typesetting system (i.e., APS, Linotron 505, Xerox 9700, IBM 6670, IBM 3800, etc.). This capability is optional. IRS is especially interested in interfacing with IBM's Script/VS (DCF) using generalized markup language. Please include experience using tapes produced using these program products.

7) All bidders must specifically cover the systems, procedures, and capabilities required in the Quality Factors for driver composition. This must include a detailed discussion of the hyphenation program, including a statement of accuracy. Include the steps taken in the justification routine (i.e., word space to preset limits, hyphenation, letterspacing, etc.), pagination routines and capabilities (specifically listing all restrictions for turn page graphics, widths and depths, and a description of the scheduling, control, and customer service systems.

3.4 Sample Submission

1) Sample submissions are required from all contractors not currently qualified as proof that the contractor can successfully produce GPO driver tapes, IRS masterfile format tapes and special looseleaf requirements. The contractor must also complete a test to demonstrate his ability to create driver tapes from a masterfile tape supplied by IRS. Those contractors who bid on producing GPO driver tapes and are not already certified for driver tape production must submit GPO driver tape with sample pages for each category on which the contractor is bidding.

2) When the contractor is ready to attempt a qualification test, he/she should contact IRS and make an appointment to pick-up manuscript copy and a masterfile tape. Sample submission pages must be consolidated on one reel of magnetic tape with a paper label on the reel giving the number of frames on the reel, and the fonts to be used. The tape will be returned to the contractor after the pages have been set at GPO. The contractor will not be reimbursed for the test tape. The sample pages will become part of the contractor's offering. When, at a later date, the contractor wishes to be certified for another category, he/she may submit another tape to prove his/her capability. The cost of running the job on a GPO phototypesetter will be paid by IRS.

SECTION 4—CONTRACT ADMINISTRATION CONTACT

1) In the event your firm is awarded a contract, please designate below the person whom the IRS may contact during the term of the contract for prompt action on matters pertaining to administration of the contract:

Name:

Title:

Address:

Phone Number:

2) Individuals named on particular delivery orders will be designated as the IRS Project Coordinator for that job under this contract. He/she will see that the Government receives all services called for in this contract and will represent the contracting officer in the technical phases of the work. However, in no event shall any understanding, agreement, contract modification, or other matter in deviation from the terms of this contract between the contractor and a person other than the contracting officer be effective or binding upon the Government unless formalized by proper contractual documents signed by the contracting officer.

SECTION 5—ORDER PROCEDURE

1) To place a job on this contract, the IRS will first determine whether or not to use a driver tape or contractor-supplied pages. IRS will then determine the category of composition and ''quality'' factors (i.e. hyphenation, justification, and page makeup) required. IRS will compare these requirements with the capabilities offered by each contractor. Those not capable of meeting the requirements will not be considered for that particular job. Then, IRS will estimate the cost of the job using the prices supplied by contractors who have bid on the category and are able to provide the capabilities required. IRS will offer each job to the lowest contractor first, the second lowest next, etc., until the job has been accepted.

2) Exception: Noncompliance with the shipping schedule or a product judged by the Contracting Officer as not meeting the requirements of the contract will be cause, and the Contracting Officer reserves the right, to withhold further offers from the contractor until he/she is judged by the Contracting Officer to have established adequate production controls,

scheduling facilities, and quality inspection procedures to fulfill the requirements.

3) The low contractor, for a specific job, and each successive next low contractor shall be obligated to accept the job except when he/she is unable to meet the shipping schedule. When the job is accepted, a Standard Form 147, Delivery Order, and a Form M-5905, Electronic Composition Work Request, will be issued. This shall obligate the contractor to complete the job on the schedule promised. No contractor shall perform work of any kind until both are received. Each contractor, upon declination of an offer, shall confirm such declination in writing. The *only* valid reason for declining an order shall be the inability to meet the shipping schedule.

4) While the delivery order (SF-147) will cover the entire job, the work request (Form M-5905) (see Attachment 1) will cover each cycle required. The contractor will be provided the white, yellow and pink copies of the work request and must return, within three working days, the pink copy of the work request with item 12 completed. The yellow copy, with item 7 completed, must be returned to IRS either with the job or within two workdays after delivery of the job. The IRS will use the yellow copy for verifying all bills submitted by contractors. Failure to return this copy may delay payment of invoices.

SECTION 6—SCHEDULE

6.1 General

1) Because this is an open-end contract, each job, except for looseleaf production, will be scheduled using the number of workdays outlined below. The schedule is based on working days beginning the day after the contractor is notified that the job and the copy are available for pickup. Delivery to Room 1550 *MUST* be made by 3:30 p.m. on the final workday.

6.2 Workdays Allowed

1) For initial keyboarding of up to 2,000,000 masterfile characters, 16 workdays will be allowed to deliver a CPO, galleys, or driver tape for galleys. One additional workday will be allowed for each additional 200,000 characters.

2) For converting a masterfile tape to a CPO, galleys or a driver tape for galleys, six workdays will be allowed.

3) For initial updates to files, 11 workdays will be allowed to deliver a CPO, galleys, or driver tape for galleys. More days will be allowed when, in the determination of IRS, updates are extensive.

4) For additional updates, six workdays will be allowed to deliver a CPO, galleys, or driver tape for galleys. More days will be allowed when, in the determination of IRS, updates are extensive.

5) For updates where up to 150 typeset pages will be generated, eight workdays will be allowed to correct the file and return (as requested on the work order) a driver tape for pages, contractor-generated pages, masterfile tape(s), and/or a revised CPO. One additional workday will be allowed for each additional 150 typeset pages.

6) After final pages are received, there are times when errors are found that were made by either the contractor or IRS. The contractor must be prepared to correct these errors within a 36-hour schedule. A 50% surcharge to the minimum charge will be allowed for IRS errors (the minimum charge will be $225 for this service).

7) An exception to the above schedules for looseleaf production (Category Five) is cited in 12.2.6.

6.3 Communication

1) It is imperative that contractors keep IRS informed of any problems that may endanger the successful completion of a job. Contractors are urged not to accept any job that they cannot complete on schedule. In the event that an order is placed with a contractor who cannot meet the contract schedule, he/she must notify IRS within 72 hours after receipt of copy of their inability to meet the schedule. *A failure to notify within 72 hours will constitute acceptance whereupon all provisions of the contract will apply with respect to such order. A failure to meet this obligation will be grounds to consider the contractor in default.*

SECTION 7—PICKUP AND DELIVERY OF MATERIAL

1) Contractors will be required to pickup and deliver all material at the Internal Revenue Service, Room 1550, 1111 Constitution Ave., N.W., Washington, DC 20224. Contractors will be called whenever material is ready for pickup. Deliveries may be made on the scheduled day between the hours of 8:15 a.m. and 3:30 p.m. Pickup and/or delivery services which cannot provide service to the specific room are not acceptable. If delivery has to be made to other locations outside the IRS building, the contractor will bill the actual delivery costs incurred. This must be supported with invoices, etc.

SECTION 8—SECURITY

1) Some jobs will be classified ''For Official Use Only'' and will be indicated as such on the work request. Release of the material to anyone not authorized by IRS is prohibited. Contractors must undertake the following measures to protect the materials sent to them:

a) all copies must be accounted for and waste sheets destroyed beyond recognition or reconstruction.

b) only contractor's authorized personnel are permitted to work on the materials.

c) the production facilities must be locked or guarded.

SECTION 9—SUPPLIES

1) For IRS work the Service will supply the contractor with a sufficient amount of machine stationery (9⅞″ × 11″ or 14⅞″ × 11″). If necessary, the contractor will be required to decollate and distribute the parts as described on the work request. It will be the responsibility of the contractor to reorder paper from IRS as required.

2) For jobs where IRS does not provide CPO paper, the contractor must be able to supply standard 9⅞″ × 11″ or 14⅞″ × 11″ printout paper in one to six parts as required. Paper may be chemical transfer or physical transfer. Carbon interleaved paper is not acceptable.

3) On jobs where a masterfile has been previously created, the Service will supply the contractor with one copy of the masterfile tape for creation of the CPO or galleys for updating.

4) The contractor will supply all magnetic tapes to be used during the term of the contract. All driver tapes will be returned to the contractor after GPO returns them to IRS.

5) To reduce the chance of losing the masterfile, IRS will maintain back-up masterfile tapes. The contractor will be required to furnish masterfile tapes each time the work request calls for them. Upon receipt of a new tape, the Service will return the old one.

SECTION 10—PROOFREADING

1) The contractor will be responsible for all proofreading. If any CPO or galley proof is supplied to IRS averaging more than two printer's errors per 7,500 masterfile characters, it may be rejected. Should proofs be rejected, the contractor will be required to reproofread, update the file, and provide a corrected CPO or galleys (or driver tapes for galleys) at no cost to the IRS. If one of GPO's phototypesetters was used to create the initial proofs, the contractor will also be responsible for the cost of the second phototypesetting at GPO. For those jobs where the contractor is required to go directly to pages, the acceptable error rate will be no more than one printer's error in 7,500 masterfile characters.

SECTION 11—DESCRIPTION OF GENERAL REQUIREMENTS AND IRS STANDARDS

11.1 General

1) The electronic composition program for IRS was developed to provide commercial sources for file creation and maintenance, composition, pagination and related services. To accomplish this, standards have been established for producing masterfile tapes which allow contractors to convert IRS masterfile tapes to their various systems and, after the job requirements have been met, to edit the file back to the IRS standard masterfile format. The basic concept is that two tapes are produced—a device dependent tape to drive GPO phototypesetters, and a device independent masterfile tape to provide file portability.

2) The electronic composition program has been expanded to include the benefits of generic coding, the generic identification of textual elements and heirarchies. This approach, in addition to enhancing the processing capabilities of text files, will provide for increased file portability, multiple uses of files and simpler mark-up. IRS will follow the guidelines established by the Graphic Communications Association's ''GenCode Primer—A Method of Generic Textual Element Identification'' and the work of the ANSI X3J6 Markup Language Working Group.

11.2 Masterfiles

1) To continue processing files created with locators and to expand into generic mark-up concepts, three types of masterfiles will be required: masterfile A which uses IRS standard locators; masterfile B which uses fully-qualified generic identifiers (FQGI); and, masterfile C which uses non-qualified generic identifiers (NQGI).

11.2.1 Masterfile A

1) Masterfile A will use established locator concepts to identify text material within the file. Locators will continue to be restricted to a two-character code (01-99, AA-ZZ) that will be preceded by a Hex BA on the masterfile.

11.2.2 Masterfile B

1) Within Masterfile B only fully-qualified generic identifiers (FQGI) will be used. No relative processing will be required, i.e., identifiers will have only one meaning. Syntax and restrictions for generic identifiers are described in 11.2.5. Additionally, text generation, symbols and reference attributes will be allowed. Contractors are required to process from the source file for each production cycle to resolve symbols and references within the text. A bid item for processing the file for this is included in Section 15.

11.2.3 Masterfile C

1) Non-qualified generic identifiers will be used in masterfile C to identify both structure and content. Text generation, symbols and reference attributes will be allowed. The source file must be processed for each job cycle to fully-qualify the generic identifiers. A bid item for this processing is included in Section 15. Text generation, symbols and reference attributes will also be allowed and the bid item for reconciling symbols and references in Section 15 would apply.

In case of conflict between any statement in any article and the standard, the standard governs.

11.2.4 IRS Standard Masterfile Format

1) Masterfile tapes will not have typographic or system codes for any composition program or typesetter.

2) A standard character set (basically EBCDIC characters) has been established (see Attachment 2). All typesetting commands, generic identifiers and text must be in the IRS Standard Character Set. Contractors must run the masterfile tape against a check program to assure that all charcters are valid and included in the standard.

3) Data must not contain hyphens except for hyphenated words.

4) The file must not contain any extraneous matter.

5) Data must be in 2500-3000 character blocks.

6) A header label must include the date prepared, job title, masterfile type and contractor's name.

7) A list must accompany the tape which indicates: the locators used in masterfile A; the FQGI's used in masterfile B; or, the NQGI's and their subsequent FQGI's used in masterfile C.

8) A paper label listing the following information must be affixed to the masterfile tape reel; “Masterfile tape,” the contractor's name, the IRS requisition number, the job title, masterfile type, and the date prepared. No other labels may be on the reels. All masterfile tapes must be on reels at least ten inches in diameter.

11.2.5 Standard Syntax for Coding Masterfiles B and C

1) During the sequential processing of the masterfile, three states can occur which affect the processing. These are content, mark-up and delimited value text.

2) Content, the most common state, includes the text of the file. Within content the program must recognize symbols (up to eight characters in length beginning with a “&” and ending with a “.”) and the call to transfer processing to the text processor (the default is “!$”). To transfer the state to mark-up, the program must recognize a mark-up open symbol (the default is a “<”). If this symbol “<” (without quotes) is not followed immediately by a generic identifier (GI) or an element termination delimiter (ETD) then it is set as content.

3) Once a valid mark-up open symbol is encountered, the state is within mark-up. The program must recognize within mark-up: the GI (an alphanumeric string of from one to eight characters of which the first must be alpha and the others are either alpha or numeric); an ETD; a value indicator (the “=”); an attribute value delimiter which changes the state to delimited value text; and, a mark-up close symbol (the default is “>”) which changes the state back to content. The attribute value delimiter ″ (double quote) is the only valid use of the ″ in the file and transfers the state to delimited value text.

4) Delimited value text is ended by another ″ which transfers the state back to mark-up. In mark-up the ″ may also be followed by a space to serve as a safety feature in the event of a keying error. If the ″ followed by a space occurs in mark-up, it is treated as a null. For example, the keying of the attribute for TITLE could be: <TITLE “short = “service contract” “security = “official use only”>

11.3 Material for Initial Keyboarding Cycle

1) Generally, typed or printed manuscript will be provided by IRS, although, on occasion, some handwritten material may be included.

2) Mark-up requirements will vary depending on which type of masterfile will be produced. In all cases the minimum amount of mark-up necessary to guide the contractor in the completion of the codification will be provided. Contractors are responsible for the correct coding of the file even if some copy is incorrectly marked-up by IRS.

3) Manuscript for masterfile A will be provided with a Linotron Locator Log (see Attachment 3) or similar instructions which clearly indicate the coding and formatting requirements of the job.

4) Masterfile B manuscript will be provided with FQGI Descriptions and Symbol Descriptions (see Attachment 4). Contractors are cautioned that manuscript copy may contain the text which will be generated from a generic identifier, symbol or reference id. This text must not be keyed and should be deleted during mark-up.

5) Manuscript for masterfile C will be provided with a NQGI Structure Definition which diagrams the relationship of the generic tags, in addition to the FQGI and Symbol Descriptions.

6) As requested by IRS, the contractor will return CPO's galleys, GPO driver tape for galleys, pages, and/or GPO driver tapes for pages.

11.4 Supplied Masterfiles

1) For files which have already been created and are on tape in the IRS masterfile format, the contractor will be supplied with a copy of the masterfile tape.

2) Depending on the masterfile type, contractors will also be supplied with the FQGI Descriptions and/or the NQGI Structure Definition or the Locator Log.

3) As requested by IRS, the contractor will return CPO's galleys, and/or GPO driver tape for galleys to be used for updating the file.

11.5 Formats

1) Format development must be authorized either on a delivery order or on the work request prior to the start of work. Contractors are cautioned against doing work on a number of formats without prior agreement. IRS will not compensate contractors for unauthorized work.

2) All format development is based on the number of

In case of conflict between any statement in any article and the standard, the standard governs.

fully-qualified generic identifiers (FQGI's). For masterfile A the locator is a FQGI.

3) Since each FQGI could have different processing requirements, the pricing will assume that they do. A single format development charge is allowed for the first set of 40 FQGI's. Each additional is priced at 1/40th the price of the format development charge.

4) Contractors are entitled to single format development charge when requested to process jobs from an IRS masterfile created by another contractor.

5) When existing formats require modifications a flat fee of $50.00 plus $5.00 per FQGI changed will be paid by IRS. Modifications include temporary or permanent format changes or duplication of an existing format with minor changes for a special application. IRS will provide modifications either by a Format Change Notification for masterfile A (see Attachment 5) or by a revised FQGI Description (see Attachment 4).

6) Contractors are required to minimize the number of formats for any masterfile A or B job. For masterfile C jobs only one format is required since the coding of textual elements is relative to their structure.

7) Most tables will not be set using a format unless the table will be updated and reissued. IRS must be consulted before a format is developed because the cost for rekeying a nonstandard table versus developing a format must be considered.

8) Some jobs may require special characters or functions which would not be practical to add to the standard character set or to define as locators or generic identifiers. Floating accents and cancelled zeroes are examples. For masterfile A jobs, these special items should be described in the same locator format and included in the Locator Log. For masterfile B and C jobs, these will be symbols included in the Symbol Descriptions.

11.6 Allowing Space for Graphic Inserts

1) IRS will designate space to be left for graphic inserts for either ''mandatory'' or ''non-mandatory'' positioning. Mandatory positioning means that the space must immediately follow (or precede in some cases) the text to which the graphic insert applies. Sometimes, it will be mandatory for the space to appear immediately following, and on the same page with a specific portion of text. Non-mandatory positioning allows the contractor to leave space at the next convenient point, such as at the bottom of the column or page, or at the beginning of the next column or page.

2) Most space to be left in single column jobs or space which spans only a single text column in multi-column jobs will require mandatory positioning. However, wherever possible, the IRS will allow non-mandatory positioning, particularly for space which spans more than one text column because of the pagination difficulty that it causes. Full pages which must be left for graphic inserts will usually be designated for ''non-mandatory'' positioning because it is undesirable to leave a short page simply because a graphic insert follows.

3) The coding of space on the masterfile will be done using a ''99'' locator followed by the phrase ''See Printed Copy'' followed by a statement indicating whether it is mandatory or non-mandatory, the size (i.e., 2 columns, 5 column inches) and the title of the graphic.

4) Some jobs may require the contractor to strip-in graphic inserts supplied by IRS. Inserts will be identified to correspond with the ''99'' coding on the masterfile. This service will be based on a per graphic charge. Contractor's bid price should include the cost of stripping graphics in place.

11.7 Computer Printouts

1) Computer printouts will vary depending on masterfile type and need. Generally, two types of CPO processing will be required: a listing of the ''source'' file; and, a formatted CPO.

2) The source file CPO must be in a two-line format where the first line contains the text and the second line contains the typesetting and control codes or the generic identifiers. Typesetting codes must be in the format shown in Attachment 6. The print line should be 6″ (60 print positions) or less and must only end on word spaces, ends of sentences, or hyphens. Characters on the printout must be upper/lower case.

3) The formatted CPO would simulate camera copy but would be output from a printer. For example, the first level head, identified as H1 in the file, could be centered, all caps and underscored on the CPO, but, when typeset, it could be 12 pt. bold, upper/lower case and flush left. No generic identifiers would appear on the CPO and all generated text, references and symbols would appear as they would in the final output. In addition to the charge for producing a CPO (15.2), and any applicable processing charges (15.11, 15.12, 15.13, 15.124, or 15.15) and 50% of the format development charge would apply.

11.8 Printout Standards

1) All printout pages must begin with the following:

a) Current Date

b) Contractor's Name

c) Job Title

d) IRS Requisition Number

2) At the end of each printout there must be a statement listing the total number of masterfile characters.

3) On the printouts, there must be an indicator printed in the right margin to indicate where changes were made for that update. During each update, the old indicators shall

be stripped from the file before indicating where the latest revisions were inserted.

4) The printout must have identifiers printed in the left margin and/or on the control line to identify uniquely the location of the data in the file at least to the level of a paragraph.

5) All special characters must print as descriptive mnemonics (i.e., ¢ might print as "cent mrk"). The mnemonics should be enclosed in parentheses with a hyphen below the parentheses.

6) Some jobs may require that the printout be a line-for-line representation of the galley proof. Based on the indication made in 11.9(4), contractors unable to meet this requirement will be excluded from just those jobs requiring this. If the contractor cannot have the change indicators appear on the CPO (see (3) above) then the contractor must supply at no cost to IRS an additional CPO, showing the changes where the CPO is not line-for-line.

11.9 Galleys

1) Galleys will only be requested when a driver tape (or the contractor's phototypesetter) will be used for the final page composition. When galleys are requested, the contractor must have the same line numbers or codes he/she requires to correct the file typeset on the galley (preferably, in the left margin) that appear on the CPO. The contractor may not transcribe the material to a CPO for file update or use only "text searches" on a Video Display Terminal system.

2) IRS may require a CPO for an initial update of the file and a galley for a later update or vice versa. The contractor must be able to update the file using either media.

3) The galley must be composed (hyphenated and justified) with proper leading. The galleys may be used for page mock-up by IRS in some instances, in which case, the line numbers or other file location codes will be left on the galley.

4) On some occasions, the IRS will require the CPO's to be line-for-line with the galleys or page proofs. Inability to comply with this requirement will cause the contractor to be excluded from the sequence of contractor selection for that particular job. The contractor should indicate here whether or not the CPO's can be produced on a line-for-line basis with the galleys. No response will cause IRS to assume that the answer is negative.

__________ Yes __________ No

11.10 Data Base Storage

1) The contractor must maintain a data base of the current edition of each job and format produced under this contract for as long as he/she has a contract with the IRS, or until he/she is informed to dispose of the file. No reimbursement will be made specifically for this service.

SECTION 12—DESCRIPTION OF COMPOSITION REQUIREMENTS

12.1 Quality Factors

1) The IRS will determine, based on each particular job, the required "quality" in the following three areas: hyphenation, justification, and page make-up. The contractor will indicate his/her capabilities in Section 16. If a job requires a quality factor the contractor cannot provide, he/she will be excluded from the sequence of contractors determination.

12.1.1 Hyphenation

1) Hyphenation capabilities listed in descending order of accuracy include:

a) One hundred percent correct hyphenation. No use of algorithms acceptable (all words properly hyphenated). Preferred hyphenation points must be used whenever possible. It is not necessary to use any particular dictionary as a source but the online dictionary must have at least 150,000 words. Homographs must be properly hyphenated. Whenever a word to be hyphenated is not found in the dictionary, it must be manually hyphenated and the word added to the dictionary.

b) Logic with an exception word dictionary. Contractors must have a sufficiently accurate program and a large enough exception word list to assure at least 95% accuracy. State the number of words in the dictionary. Also state how new words may be added (include with subsection 3.3 submission).

c) Logic only. At least 90% accuracy must be maintained.

2) If there is no hypehenation in the job, no contractor will be excluded. All contractors who can provide hyphenation capabilities equal to, or better than what is required will be considered in the sequence of contractors.

12.1.2 Justification

1) Justification capabilities include:

a) Word space only

b) Word space and letter space

c) Letter space only

d) Flush left (ragged right):
   with hyphenation
   without hyphenation

12.1.3 Page make-up

1) Page make-up refers specifically to the placement of text, heads, tables and other typeset "exhibit" material on the page.

a) All typeset material must be set "in place" with space left for strip-ins at any point.

b) Space for strip-ins must be left on a page according to some contractor/system predefined rules (corners, top, bottom etc.).

12.2 Composition Categories

12.2.1 General

1) This contract recognizes that all composition programs are not capable of doing all types of composition, and even within different types of composition, there are differences in the subjective qualities of composition. For example, it is not necessary to have a hyphenation program for composition of telephone directories. To accommodate the inherent differences that exist in composition programs, five general categories of composition have been identified.

12.2.2 Category One

1) Generally, this category includes straight matter work with tables and heads which span only a single text column (see Attachment 7). Folios, running heads, foot references, etc., will normally fall outside the text area, although, in some cases, a short column may be required throughout a publication for page folios and/or a running head or foot.

12.2.3 Category Two

1) This category will include text work more difficult than Category One (see Attachment 8). The work consists of two columns with heads and/or graphics spanning one or both columns and inserted without necessitating any cut and paste. Work may be required to be paged according to a mock-up or dummy prepared by IRS.

12.2.4 Category Three

1) Includes text work more difficult than that which falls under Categories One and Two (see Attachment 9). Work will consist of three or more columns of text. Heads, tables and space for graphic inserts may span across one or more columns without necessitating any cut and paste. Work may be required to be paged according to a mock-up or dummy prepared by IRS.

12.2.5 Category Four

1) Includes publications that are primarily statistical tabular material. This includes box heads, stubs, foot references, braces, etc. (See Attachment 10).

12.2.6 Category Five

1) This category is for looseleaf-type publications. An example of looseleaf material is shown in Attachment 11. Contractors interested in bidding on this area are cautioned to first see and understand the intricacies of the looseleaf material. The contractor must be able to update existing publications on a leaf-by-leaf basis. To accommodate the differing composition programs used by various contractors and to obviate the need to reprint the looseleaf product with each change of contractors, each leaf in the book must begin with a new paragraph or head. However, paragraphs may be split between leaves when they contain more than 30 lines.

2) Looseleaf publications require a highly specialized type of composition, since most general purpose composition programs cannot selectively compose pages and "squeeze" a few more lines onto a leaf to reduce the number of pages set. For example, the Internal Revenue Manual is set 8/10, but the composition program must be able to reduce that to 8/8.5 if it can save production of another leaf in the update cycle. The use of "point" page numbers and "half" pages is only acceptable when the material cannot be made to fit on the next or preceding leaf. Whenever material is deleted, the pagination program should carry the remaining material forward, leaving only the last page short. Whenever material is added, the contractor should attempt to fit it using the following sequence or combination of solutions:

a) Compress it on the leaf by reducing the leading down to one-half point and the space around the heads to one-half their normal point value.

b) Compress it on the previous or next leaf, using the criteria in (a) above. Choose the leaf on which the added material will most readily fit.

c) Issue "point" pages following the guidelines in paragraph (8) below. Only the last page may be set short.

3) As a general rule, each change to the file should be independent of other changes. However, at times, changes may be close to each other and it may be more practical to link the changes together to eliminate short and/or point pages. For example, it is possible for a transmittal to add a half-page of text to leaf one and remove a half-page from leaf three. In this instance, the two should be linked so the runover text from leaf one will set on leaf two and the text from leaf two will flow over and fill up leaf three. All the changes would therefore be printed without having to issue both a short page and a point page. Each transmittal will have to be individually analyzed by the contractor to determine the best way to issue the changes.

4) For original material, the contractor will be furnished manuscript and will return either galleys or pages. The contractor is responsible for *all* proofreading. Excessive errors will result in removing the contractor from the contract. The initial keying of looseleaf publications is covered by standard keyboarding, correction and composition category charges. Special pricing provisions have been made for updating previously published material.

5) Update material will be provided as single-column

In case of conflict between any statement in any article and the standard, the standard governs.

manuscript of the current printed material using complete “TRIDOC’s”. A TRIDOC is the material from one head to the next. The contractor must update the file and return either pages or a driver tape of the affected pages (depending on whether or not a wavier has been granted by GPO). The contractor must also complete the “Removal and Insertion” instructions on the transmittal and update the list of current pages when requested.

6) The contractor must be able to extract and compose the Table of Contents, when requested.

7) On occasion, the IRS will request a complete reissuance. In which case the contractor will recompose the file. At this time the file will be repaginated and renumbered so that all “point” pages, short pages, etc., are eliminated.

8) Page numbering is sequential with some numbers skipped to allow for expansion. Generally, four or five numbers are skipped prior to exhibits. The exhibits start on a new odd page. Insertion of additional material is done with “point” numbers. The point numbers will start with even page numbers (i.e., 1, 2, 3, 4, 4.1, 4.2, 4.3, 4.4, 4.5, 5, 6, etc.). Whenever the next page number is not in the normal sequence (as is the case between 4.5 and 5 in the preceding example), a “Next page is ______” statement must be set on the bottom of the page. The contractor is responsible for all pagination determinations.

9) The contractor will be placed on the distribution list to receive a printed copy of the material to allow for text pagination.

10) A double set of prices are requested for this category to allow those products that are not covered under a waiver to be run on the GPO’s phototypesetters.

11) The masterfile is a standard IRS masterfile with no pagination information. All references to material in the printed text will be by TRIDOC number. The IRS will request periodic masterfile tapes of the file, and will also require tapes of just the new and revised “TRIDOC’s.” These tapes will be used for an information retrieval system.

12) IRS will not schedule more than one transmittal (update) to an individual file at one time. For example, if issuance number one is in process, then IRS will schedule issuance number two at least six days after the scheduled completion date of issuance number one.

13) Looseleaf manual issuances will be produced from manuscript on the following special schedule: This schedule assumes that the contractor is notified that the text will be updated at least 24 hours prior to the beginning of the workday count. This will allow scheduling and file loading prior to receipt of copy.

a) 1-32 typeset pages in five workdays

b) 33-96 typeset pages in eight workdays

c) 97-200 typeset pages in ten workdays

d) over 200-500 typeset pages in twelve workdays

e) Over 500 typeset pages—one additional workday will be allowed for each additional 100 pages

14) A surcharge can be indicated in items 25a and 25b of the schedule of prices as reimbursement for the expedited schedules required in paragraphs (11) and (13) above.

15) Some of the pages will require change bars (|) and scratch marks (#) in the margin. When required, they will be clearly indicated on the manuscript. The changes bars must be deleted when the page is recomposed because of either an update or a reissuance. The scratch marks will remain where indicated in the text. The scratch marks are currently used (but not limited) to indicate restricted material that is not issued to the public reading rooms.

SECTION 13—DESCRIPTION OF COMPOSITION RELATED REQUIREMENTS

13.1 Special Options

1) On occasion, the Service will have jobs that will require special computer processing. These may include sorts, extracts, etc. This category will allow the service to purchase “off-the-shelf” capabilities from contractors.

2) This category will be used only on an existing masterfile. Some computer processing will be done to produce another masterfile which will then be processed using another category. No keyboarding, updating, composition, etc., is provided for in this category as all such operations would be done using other provisions within the contract.

3) Three charges are provided for:

a) A one-time setup charge

b) A flat charge for each time a job is processed

c) A per thousand rate for processing the file to create the new masterfile tape

4) A series of options are listed in Section 15 for which the bidder may quote prices. The bidder must supply detailed provisions, capabilities etc., for each option on which he bids (include with subsection 3.3 submission). The contractor may also add additional options, and may quote prices in the same format as those provided. When more than one function is required for a particular job (i.e., Extract and Sort), then the largest one-time setup charge will be used plus 50% of each additional function’s setup charge. This charge will only be allowed once, regardless of how many times the job is processed. The flat charge will be allowed for each time a job is processed. When multiple functions are required, the full flat charge will be allowed for each option. The per thousand rate will also be the combination of charges of all options required.

5) Most files that will be used for the Special Option Section will be created just for the purpose of using these computer options (data base or structured-type files), however, others may be existing text-type files. All provisions for the IRS standards must still be met, even if the file is

In case of conflict between any statement in any article and the standard, the standard governs.

all fixed field data. All fields and data elements must be identified with locators or tags and use the IRS standard character set.

6) Most structured data bases will be created using an input form. The contractor may be consulted for suggestions on the design of the form. An output questionnaire may be required for use in updating the file. The questionnaire form will be supplied by IRS as a continuous form or may be required to be produced by the contractor on a "9700-type" or "3800-type" device. The charges for producing the questionnaire on IRS-supplied CPO will be the same as producing a standard CPO. If the questionnaire is to be produced using an electronic printing device, charges are included in 15(2)(d). If a questionnaire is used to update or create a file, a 50% surcharge will be allowed for the charge per change (item 4 of Section 15).

7) Additionally, the Service may have, on occasion, jobs which require copy preparation, transcription, and/or editorial services either by the contractor or online to the contractor's system using IRS personnel. The cost of these services will be negotiated between the IRS and the contractor on a per job basis.

13.2 Readability Option

13.2.1 General

1) At IRS option, a readability test may be needed using grade level analysis. If the contractor has the means to do this kind of test, please indicate by completing the prices in item 29 of Section 15. If this option is required, those contractors not able to do the testing will not be included in the analysis to determine low bidder.

13.2.2 Scope of Work

1) The contractor will determine the readability grade level of publications on the masterfile. The IRS will inform the contractor of the material to be analyzed.

2) The readability assessment will be done using approximately 200-word "samples" or segments. The readability levels will be determined by using the following formulas: The Fry Graph, the Dale-Chall, the Flesch, and the Harris-Jacobson word lists.

3) For each segment of the text analyzed, the contractor will list the number of words per segment, the number of sentences per segment, the number of words per sentence, and the number of words in each segment which do not appear on the Dale-Chall and Harris-Jacobson word lists. The Key Word in Context printout must be in the same format and contain like information as shown in Attachment 12. Words to the right of the vertical line on the printout are considered unfamiliar words using the Harris-Jacobson and Dale-Chall word formulas.

4) The contractor will indicate on a printout of the text, the words that the contractor determines would add to an increase in the readability level, difficulty, or some comprehension criterion, based on the Dale-Chall and Harris-Jacobson word lists. Modifications to these word lists will be made to accommodate certain technical words which are required for the text. These modifications will vary from job to job, but an example follows:

a) The words Form, Schedule, Publication, Internal, Revenue, and Service will be treated as familiar words when they appear capitalized in our material;

b) The months of the year are to be treated as familiar words;

c) All numbers, all letters, and the names of all states will be treated as familiar words;

d) All relatives not appearing on either the basic Dale-Chall or Harris-Jacobson word lists will be treated as familiar words; and,

e) The titles of all our Taxpayer Information Publications will be treated as familiar words.

5) The contractor will furnish a readability analysis in three computer printouts containing the information mentioned above. The first printout will contain an analysis of the material using the unadjusted Fry, Dale-Chall, Flesch, and Harris-Jacobson formulas. This printout will reflect the basic Dale-Chall and Harris-Jacobson word lists, including the inflected forms of the words on those lists. The second printout will contain an analysis of the material using only the adjusted Dale-Chall and Harris-Jacobson formulas by incorporating the modifications listed in (4) above. The third printout will be the Key Word in Context listing which will also treat the modifications in (4) as familiar words.

SECTION 14—EXPLANATION OF CHARGES

14.1 General

1) Most prices are based on the count of the characters on the masterfile. This allows a standard for pricing because of the many keyboarding and coding systems used by contractors.

2) There is a charge per change to allow for the cost of locating a change on the file and keying the codes required by the contractor's system to effect the change. The actual keying of the characters changed is charged in a separate item and is based on the number of masterfile characters changes. A deletion, on the other hand, is a single charge regardless of the size of the deletion.

Note: The contractor will be responsible for any costs incurred by the IRS to re-run any job on a GPO phototypesetter that is caused by mistakes that are clearly and solely the fault of the contractor. These costs will not exceed $2.50 per page.

In case of conflict between any statement in any article and the standard, the standard governs.

3) A minimum charge of $200 will be allowed for each initial keyboarding cycle (whenever items 1 and 2 or 8 or 9 of Section 15 do not total $200, the minimum charge of $200 will be allowed). A minimum charge of $150 will be allowed for each subsequent update cycle (whenever the costs of items 2 through 26 do not total $150, a minimum charge of $150 will be allowed), except as noted in 6.2(6) when a $225 minimum is allowed for special expedite service. This does not include a cycle required solely to correct contractor errors. Lower minimum charges may be negotiated between the Service and the contractor for program-type work. The reduced minimum charge will *not* be used in the basis of award for determining the low bidder for the job.

4) All prices are mutually exclusive. For example, tabular material cannot be billed as text keying item 1, and typeset characters item 10b.

14.2 Composition Charges

1) The charges for page composition in each category are divided into two parts. First is the charge for composition (based on the number of masterfile characters composed) and the second is the charge per page. The composition charge (just as the galley charge) is for the hyphenation and justification of the file, while the per page charge is for the actual page make-up. The contractor should note that IRS will *not* run the composed tape on GPO phototypesetters for the contractor to check the page make-up. IRS expects to receive a tape from the contractor that will set pages ready for printing. Any cost incurred checking the composition and page make-up must be included in the bid price for the two items.

2) IRS expects that on occasion, author's alterations will be made to a job already composed and in page form. When this occurs, the contractor *must* be able to correct the file and produce a driver tape with just those pages affected by the alterations. It would also be acceptable to produce a driver tape of the entire job with a list of those frames changed. IRS will only pay for the composition and page charges for those pages changed (or the minimum charge—whichever is larger).

14.3 Seasonal Price Discounts for Initial Keyboarding

1) In an attempt to assure the IRS of consistent service at the best possible price, contractors may indicate seasonal price discounts for initial keyboarding. These discounts are to be stated as a percentage of the keyboarding price (item 1) for all work sent to the contractor during the month in which a discount applies. Each bidder can offer discounts for any month(s) of the year. The discount for any given month applies to all work sent to the contractor during that month, even though the work may not be performed until the following month. Discounts will be considered when determining the sequence of contractors. Unless otherwise indicated, these discounts will be considered positive numbers and result in a reduction of the bid price.

14.4 Special Publishing-Related Computer Programs

1) Occasionally, jobs ordered under this contract may require special publishing-related computer programs to be written. For example, programs may be required to edit data tapes supplied by IRS not in the IRS masterfile format. The cost for development of these programs will be negotiated between IRS and the contractor. The program will remain the property of the contractor. The cost of developing the basic programs to output IRS masterfile format tapes is not reimbursed by IRS. —See also subsection 13.1, Special Options.

14.5 Direct to Page Surcharge

1) A surcharge can be bid in item 7 for extra proofing and updating required when in the same cycle the contractor is required to do initial keyboarding from manuscript or initial updating to an existing file for galleys or a CPO and supply either a GPO driver tape for pages or pages off the contractor's phototypesetter (if a waiver has been granted). If there is more than one printer's error in 7,500 characters, the pages or the driver tape may be returned to the contractor for reproofreading and updating and the surcharge will not be allowed. The contractor will be responsible for all costs associated with this update cycle.

14.6 Contractor Phototypesetting

1) Government agencies must obtain GPO's permission (via waiver) to buy phototypeset pages directly from a commercial contractor. Since IRS does have permission to buy some typesetting commercially, this contract will include a charge for contractor-supplied pages. These charges do not apply to any phototypesetting done by the contractor for his/her own proofing purposes before the creation of driver tapes. No contractor can bid on just providing pages from his/her typesetter. Any phototypeset pages ordered directly from the contractor must be produced as full-page output from the phototypesetter. No cut-and-paste for page make-up purposes is allowed after phototypesetting. If stabilization paper is used, it must be fixed and washed.

4.7 Tabular Composition

14.7.1 General

1) Tabular Composition in text—For each text job, one tabular layout consisting of up to five columns falling within one text column will be designated as the ''standard'' tabular layout. Tabular planning and tabular keyboarding charges

are not applicable to tables which can be made to conform to this standard. However, any table with downrules is considered "non-standard."

2) The tabular keyboarding charge, pricing item 10b, is applied in lieu of pricing item 1 for all non-standard tables in text and for Category 4 tabular keyboarding.

3) The contractor must provide a "galley-type" photocomposed proof of all non-standard tabular work in lieu of a CPO. This proof must have adequate space between the lines to allow for author's alterations. The contractor must also provide a photocopy of the page, since the photocomposed proof paper is not suitable for marking changes. A 50% surcharge to pricing item 9 will be allowed to cover the added cost of material.

14.7.2 Table Planning

1) Table planning will usually not be used when the IRS sets numerous tables in the same design, updates the tables for subsequent reissue, or plans to transfer the tables to another supplier for future reissue. Instead, a format for the particular tables will be designed and locators will be assigned to the data elements. In these instances, keying will be priced under item 1. The primary consideration will be the total overall cost when IRS decides to use either table planning or format development.

2) Tables are made up of rows and columns. Usually, the first column is a stub. A table can consist of an infinite number of rows and an infinite number of columns.

a) When all the columns cannot be set on one page, the stub (first column) is repeated on a second page. Additional pages are set until all columns have been set. The width of a column is based on (1) the width of the column head, (2) the width of the largest number in the column, and (3) an adjustment for page balancing.

b) When all the rows cannot be set on a single page, the column head is repeated on the second page along with the required "Continued" information and the stub continues. Additional pages are set until all rows have been set. The leading between rows should be that specified by IRS, but must be adjusted for good page fit and to balance the pages (widow conditions, etc.)

c) When both the rows and the columns cannot fit on one leaf (items (a) and (b) above), then the combination of the two would apply until all rows and columns are set.

d) When the number of entries in the stub set less than one-half the page depth and the number of columns in the head exceeds the page width, then, below on the same page, the stub should be repeated and the columns continued. See example in Attachment 13. Conversely, when the stub depth exceeds the page depth and the number of columns in the head set less than one-half the width, then on the same page and to the right, the head is repeated and the stub continued. See example in Attachment 14.

SECTION 15—SCHEDULE OF CHARGES

Note: Unless otherwise indicated by the contractor, all discounts will be considered positive.

1) File creation (includes mark-up, keyboarding and proofreading, but not CPO's) per thousand masterfile characters excluding item 10)b):

	Masterfile A/B	Masterfile C
a) Jobs with 400,000 characters or less:	$________	$________
b) Jobs with over 400,000 to 2,000,000:	$________	$________
c) Jobs with over 2,000,000 characters:	$________	$________

d) If any seasonal discounts of the keyboarding charge are desired, indicate the discount percentage for each applicable month:

Oct _____%	Jan _____%	Apr _____%	Jul _____%
Nov _____%	Feb _____%	May _____%	Aug _____%
Dec _____%	Mar _____%	Jun _____%	Sep _____%

2a) Charge per thousand masterfile characters for producing a computer printout (CPO) on:

a) Paper supplied by IRS: ....$________

b) One to three part paper supplied by contractor: ....$________

c) Four to six part paper supplied by contractor: ....$________

2b) Charge per thousand characters for producing a CPO for electronically printed questionnaires (13.1.6) ....$________

3) Charge per deletion excluding item 22 (regardless of size)

a) for text ....$________

b) for nonstandard tabular ....$________

4) Flat charge for each change or addition to the file excluding item 23 (covers the cost of locating the change or addition in the file and keyboarding the control codes required)

a) for text ....$________

b) for nonstandard tabular ....$________

5) Charge per thousand masterfile characters for characters changed or added to the file: ....$________

In case of conflict between any statement in any article and the standard, the standard governs.

6) Charge per thousand masterfile characters for production of a masterfile tape (includes final and duplicate tapes): ....$________

7) Surcharge to total job for extra proofing and updating required when going directly to final pages from manuscript copy. Charge per thousand masterfile characters: ....$________

8) Galleys—charge per thousand masterfile characters for producing a GPO driver tape for galleys (includes hyphenation and justification): ....$________

9) Galleys—charge per thousand masterfile characters for galleys produced (includes hyphenation and justification of each galley and a "plain paper" photocopy of all Category Four work). (When double-spaced galleys are requested, a 50% surcharge to this charge will be allowed to cover added material costs): ....$________

10) Tabular—Note: table planning and tabular keyboarding charges for "standard" tabular layouts (see 14.7) are not allowed. Tabular work or "standard" tabular utilizing a format must be charged under item 1. Failure to set any table (except a "standard" one) in a format will make the table unupdatable and only a locator 99 will appear in the masterfile.

- a) Charge per column for table planning (stub counts as one column) with a minimum charge of 8 columns per table: ....$________
- b) Charge per thousand *typeset* characters for keyboarding tabular material: ....$________

11) Charge per thousand masterfile characters to convert non-qualified generic identifiers to fully-qualified generic identifiers: ....$________

12) Charge per thousand masterfile characters to reconcile symbols and references (forward only): ....$________

13) Charge per thousand masterfile characters to reconcile symbols and references (forward and backward): ....$________

14) Charge per thousand masterfile characters to do both (11) and (12) above: ....$________

15) Charge per thousand masterfile characters to do both (11) and (13) above: ....$________

16) Category One Composition—
date services available: .... ________

A) Contractor's phototypesetter — B) GPO driver tapes

- a) Charge per thousand masterfile characters for composition: ....$________ $________
- b) Charge per page:
  - 1) Image size to 5¼" x 8½": ....$________ $________
  - 2) Image size from 5¼" x 8½" to 8½" x 11": ....$________ $________

17) Category Two Composition—
date services available: .... ________

- a) Charge per thousand masterfile characters for composition: ....$________ $________
- b) Charge per page:
  - 1) Image size to 5¼" x 8½": ....$________ $________
  - 2) Image size from 5¼" x 8½" to 8½" x 11": ....$________ $________

18) Category Three Composition—
date services available: .... ________

- a) Charge per thousand masterfile characters for composition: ....$________ $________
- b) Charge per page:
  - 1) Image size to 5¼" x 8½": ....$________ $________
  - 2) Image size from 5¼" x 8½" to 8½" x 11": ....$________ $________

19) Category Four Composition—
date services available: .... ________

(Charges for planning and keyboarding are covered under items 10a and 10b above.)

- a) Charge per thousand typeset characters for composition:
  - 1) 400,000 characters or less: ....$________ $________

In case of conflict between any statement in any article and the standard, the standard governs.

   2) 400,001 to 2,000,000 characters: ....$________ $________
   3) For jobs over 2,000,000: ....$________ $________
- b) Charge per page:
   1) Image size to 5¼" x 8½": ....$________ $________
   2) Image size from 5¼" x 8½" to 8½" x 11": ....$________ $________

20) Total one-time charge for format development by contractor: ....$________

21) Charge for stripping-in per graphic. Paste-up material will be furnished ready for stripping in place. Contractor will determine the space to leave for each graphic. Graphics will be centered side to side on pages: ....$________

22) Flat charge for each deletion to file (Looseleaf Only) applies to update of previously published material and in lieu of item 3): .....$________

23) Flat charge per change or addition to file (Looseleaf Only) (applies only to updates of previously published material and in lieu of item 4): ....$________

24) Cost per page (Looseleaf Only) (applies only to updates of previously published material and in lieu of item 16b, 17b, or 18b):

	A) Contractor's phototypesetter	B) GPO driver tapes
a) Image size to 5¼" x 8½":	$________	$________
b) Image size from 5¼" x 8½" to 8½" x 11":	$________	$________

25) Single charge for preparing both the remove-and-insert instructions and the list of current pages for each transmittal (keying, composing and page make-up charges are also applicable): ....$________

26) Charge for a computer pass to generate a Table of Contents (page make-up charges are also applicable): ....$________

27) Since Internal Revenue Manual jobs demand expedited schedules (see 12.2.6(13), a surcharge can be indicated as a percentage of the standard pricing total for this work:

- a) Surcharge for IRM issuances: ________

28) Special Options Category

- a) Extract (and pick up page numbers):
   1) One-time charge: ....$________
   2) Flat charge per cycle: ....$________
   3) Charge per thousand masterfile characters: ....$________
- b) Reorder elements (change "title, page number" to "page number, title"):
   1) One-time charge: ....$________
   2) Flat charge per cycle: ....$________
   3) Charge per thousand masterfile characters: ....$________
- c) Sort (Alphabetical, Numerical, Multi-Level, Key-word-in-context Indexes, etc.):
   1) One-time charge: ....$________
   2) Flat charge per cycle: ....$________
   3) Charge per thousand masterfile characters: ....$________
- d) Redundancy Removal (insert commas, etc.—usually done after sorts for indexes):
   1) One-time charge: ....$________
   2) Flat charge per cycle: ....$________
   3) Charge per thousand masterfile characters: ....$________
- e) Global changes to the file (change one character string to another):
   1) One-time charge: ....$________
   2) Flat charge per cycle: ....$________
   3) Charge per thousand masterfile characters: ....$________
- f) Conversion of an all uppercase file to an upper/lower case file:
   1) One-time charge: ....$________
   2) Flat charge per cycle: ....$________
   3) Charge per thousand masterfile characters: ....$________

29) Readability Option—the charge for readability analysis shall be based on the price per thousand masterfile

In case of conflict between any statement in any article and the standard, the standard governs.

characters in the file. The price covers the specifications detailed in subsection 13.2 and includes the cost of furnishing the three printouts for each job.

a) Charge per thousand masterfile characters: ....$________

b) One-time programming charge per format: ....$________

30) Other Charges:

a) Charge per page for providing photocopy proofs of material (Xerox, etc.) except as included in item 9 for galleys: ....$________

SECTION 16—QUALITY FACTORS CAPABILITIES

1) Composition for Categories One, Two, Three and Five. Contractors will indicate their capabilities for:

a) Hyphenation (check all options available)

__________ 100% correct hyphenation

__________ Logic with exception word lookup

__________ Logic only

Justification (check all options available)

__________ Word space only

__________ Word space and letter space

__________ Letter space only

__________ Flush left (ragged right) with hyphenation

__________ Flush left (ragged right) without hyphenation

__________ Others (specify):

c) Page make-up (check all options available)

__________ All material set in place with space left for strip-ins

__________ Space left according to predefined rules

SECTION 17—DEFINITIONS

1) To avoid the possibility of confusion, the following definitions will apply for this contract:

Author's Alteration (AA) Any change, addition or deletion made to the manuscript after the initial keyboarding.

Change or Addition A single move or additon of word(s). May be a code, word, phrase, sentence, paragraph, or block of continuous reading matter inserted at one point. It will be classified as author alteration (AA) only when appropriate.

Character A single keystroke needed to produce an output character. It could be a letter, number, word space, or command code.

Codification Markup of copy for typesetting; monetary charges will be included in keyboarding quotation.

CPO Computer Printout. It is the readable form of the file; a listing of the file provided for proofreading and/or file updating.

Deletion An elimination taking effect at a single point after the initial keyboarding. It is also classified as an AA when appropriate.

Driver Tape Magnetic control tape created by contractor's composition program that drives GPO phototypesetters, currently the Videocomp 500.

Format A coding structure which defines all the typographic specifications for each element (head level, indention level, point size, etc.) in a specific page layout.

GPO Government Printing Office

Internal Revenue Manual A looseleaf publication which provides the internal operating procedures for IRS employees.

Linotron Locator Log Description of a format: i.e., a compilation of locators for setting a particular job. Assigned one format number.

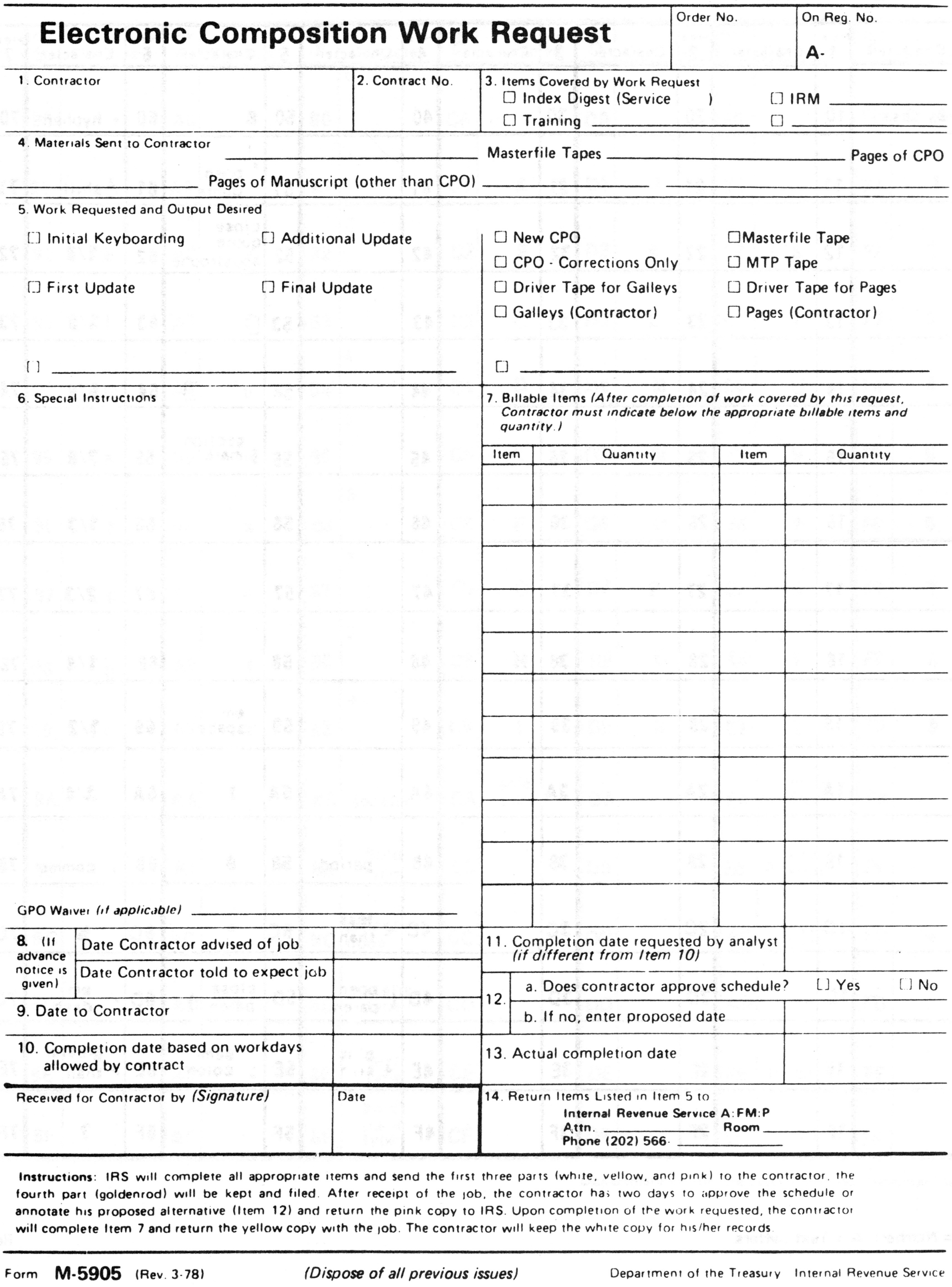
Tabular A table is a systematic arrangement of words, numbers, or signs or combinations of them, as in parallel columns, to exhibit a set of facts or relations in a comprehensive and compact form. Whenever a publication contains data which does not conform to its established format, there may be some question as to whether the data is considered to be tabular, as opposed to requireing a special format development. Work should be billed as tabular composition whenever the above definition applies. Whenever the contractor is in doubt, the IRS should be consulted.

Tridoc This term defines the smallest updatable unit contained in a looseleaf manual. It includes any heading in the IRM and all text following that heading up to, but not including, the next heading.

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

ATTACHMENT 1 - FORM M-5905, ELECTRONIC COMPOSITION WORK REQUEST



Electronic Composition Work Request

Order No.

On Reg. No. A-

1. Contractor

2. Contract No.

3. Items Covered by Work Request
☐ Index Digest (Service)
☐ IRM ________
☐ Training ________
☐ ________

4. Materials Sent to Contractor
________ Masterfile Tapes ________ Pages of CPO
________ Pages of Manuscript (other than CPO) ________ ________

5. Work Requested and Output Desired
☐ Initial Keyboarding
☐ Additional Update
☐ First Update
☐ Final Update
☐ ________
☐ New CPO
☐ Masterfile Tape
☐ CPO - Corrections Only
☐ MTP Tape
☐ Driver Tape for Galleys
☐ Driver Tape for Pages
☐ Galleys (Contractor)
☐ Pages (Contractor)
☐ ________

6. Special Instructions

GPO Waiver *(if applicable)* ________

7. Billable Items *(After completion of work covered by this request, Contractor must indicate below the appropriate billable items and quantity.)*

Item	Quantity	Item	Quantity

8. (If advance notice is given)	Date Contractor advised of job	
	Date Contractor told to expect job	
9. Date to Contractor		
10. Completion date based on workdays allowed by contract		
Received for Contractor by *(Signature)*	Date	

11. Completion date requested by analyst *(if different from Item 10)*

12. a. Does contractor approve schedule? ☐ Yes ☐ No
b. If no, enter proposed date

13. Actual completion date

14. Return Items Listed in Item 5 to
Internal Revenue Service A:FM:P
Attn. ________ **Room** ________
Phone (202) 566- ________

Instructions: IRS will complete all appropriate items and send the first three parts (white, yellow, and pink) to the contractor, the fourth part (goldenrod) will be kept and filed. After receipt of the job, the contractor has two days to approve the schedule or annotate his proposed alternative (Item 12) and return the pink copy to IRS. Upon completion of the work requested, the contractor will complete Item 7 and return the yellow copy with the job. The contractor will keep the white copy for his/her records.

Form **M-5905** (Rev. 3-78) *(Dispose of all previous issues)* Department of the Treasury Internal Revenue Service

IRS-P-83-5   47   74

ATTACHMENT 2 - IRS STANDARD CHARACTER SET

Hex 0	Character	Hex 1	Character	Hex 2	Character	Hex 3	Character	Hex 4	Character	Hex 5	Character	Hex 6	Character	Hex 7	Character
00	By pass	10		20		30		40		50	&	60	- hyphen	70	
01		11		21		31		41		51	' open quote	61	/ shill	71	
02		12		22		32		42		52	close quote apostrophe '	62	1/8	72	° degree
03		13		23		33		43		53	[	63	3/8	73	prime
04		14		24		34		44		54	]	64	5/8	74	double prime
05		15		25		35		45		55	§ section mark	65	7/8	75	- en dash
06		16		26		36		46		56		66	1/3	76	paragraph mark
07		17		27		37		47		57		67	2/3	77	en space
08		18		28		38		48		58		68	1/4	78	
09		19		29		39		49		59	em space	69	1/2	79	
0A		1A		2A		3A		4A		5A	!	6A	3/4	7A	: colon
0B		1B		2B		3B		4B	period	5B	$	6B	, comma	7B	
0C		1C		2C		3C		4C	< less than	5C	*	6C	%	7C	
0D		1D		2D		3D		4D	(open paren	5D	close paren)	6D	- em dash	7D	
0E		1E		2E		3E		4E	+ plus sign	5E	; semi-colon	6E	> greater than	7E	
0F		1F		2F		3F		4F		5F		6F	?	7F	= equal sign

X = Number: A = Text Letters

Rev. 2/15/77

Attachment 2 - Continued

Hex 8	Character	Hex 9	Character	Hex A	Character	Hex B	Character	Hex C	Character	Hex D	Character	Hex E	Character	Hex F	Character
80		90		A0		B0	0	C0		D0		E0		F0	0 zero
81	a	91	j	A1		B1	1	C1	A	D1	J	E1		F1	1
82	b	92	k	A2	s	B2	2	C2	B	D2	K	E2	S	F2	2
83	c	93	l	A3	t	B3	3	C3	C	D3	L	E3	T	F3	3
84	d	94	m	A4	u	B4	4	C4	D	D4	M	E4	U	F4	4
85	e	95	n	A5	v	B5	5	C5	E	D5	N	E5	V	F5	5
86	f	96	o	A6	w	B6	6	C6	F	D6	O	E6	W	F6	6
87	g	97	p	A7	x	B7	7	C7	G	D7	P	E7	X	F7	7
88	h	98	q	A8	y	B8	8	C8	H	D8	Q	E8	Y	F8	8
89	i	99	r	A9	z	B9	9	C9	I	D9	R	E9	Z	F9	9
8A		9A		AA		BA	locator	CA	small caps	DA		EA	- minus sign	FA	
8B		9B		AB		BB	bold type	CB		DB		EB	x times sign	FB	
8C		9C		AC		BC	precedent code	CC		DC		EC	÷ division sign	FC	e
8D		9D		AD	format	BD	italic type	CD		DD		ED	±	FD	
8E		9E		AE		BE	text type	CE		DE		EE	by pass	FE	
8F		9F		AF		BF	bold italic type	CF		DF		EF	∓	FF	

ATTACHMENT 3 - LINOTRON LOCATOR LOG

Linotron Locator Log

Format Number

Page

Date

Locators	Point Size	Leading	Line Length	Locator Leading Top	Locator Leading Bottom	Prim. Indent	Second Indent	Split	Grid Slot #	Type Face	Justification *	Line Start	Remarks
01													
02													
03													
04													
05													
06													
07													
08													
09													
10													
11													
12													
13													
14													
15													
34													
35													
36													
37													
38													
39													
40													
41													
42													
43													
44													
45													
46													
47													
48													

* Justification

L = Left J = Left & Right

R = Right C = Centered

Prepared by

Form **M-5840** (Rev. 5-74) **Page 1**

Department of the Treasury - Internal Revenue Service

ATTACHMENT 4 - FQGI AND SYMBOL DESCRIPTIONS

FQGI Descriptions	Format:	Status: ☐ new ☐ revised

Tag name:	Formatting instructions: point size: leading: indent - primary: secondary: typeface: justification:
Text generation: ☐ from reference attribute ☐ sequential: ☐ alpha ☐ numeric ☐ text:	Processing instructions:
Tag name:	Formatting instructions: point size: leading: indent - primary: secondary: typeface: justification:
Text generation: ☐ from reference attribute ☐ sequential: ☐ alpha ☐ numeric ☐ text:	Processing instructions:

Symbol Descriptions	Document:	Format:

Symbol:	Processing instructions:

ATTACHMENT 5 - FORMAT CHANGE NOTIFICATION

FORMAT NO.

Linotron Format Change Notification

To:
☐ Internal Revenue Service, A:FM:P, Rm. 1239, Att: ____________
☐ G.P.O., Electronic Printing Division, Att: ____________
☐ I.R.S. Contractor ____________
☐ Other: ____________

From:
☐ I.R.S.
☐ G.P.O.
☐ I.R.S. Contractor ____________
☐ Other ____________

1) Type of Change

Locators	Point Size	Leading	Line Length	Locator Leading		Prim. Indent	Second Indent	Leader from Bottom/Top	Grid Slot #	Type Face	Justification	Line Start	Remarks
				Top	Bottom								

2) Nature

☐ Make change immediately
☐ Wait until ____________ to make change
☐ Permanent change
☐ Temporary change

3) Explanation

4) Attachments:
☐ New parameter listing with changes circled
☐ Graphic or written instructions
☐ Other: ____________

Form **M-6101** (2-74)

Internal Revenue Service — Department of the Treasury

ATTACHMENT 6 - IRS PRINTOUT STANDARDS

IRS Printout Standards

Typesetting Code	Master File Hex	Printout
a) Bold Type	BB	AAA AAA BLD
b) Italic Type	BD	AAA AAA ITL
c) Text Type	BE	AAA AAA TXT
d) Bold Italic Type	BF	AAA AAA BIT
e) Locator (item)	BAXX	AAA AAA XX
f) Paragraph	BCD7	AAA P
g) End of Frame	BCC5	AAA End of Frame
h) Format Number	AFXXXX	AAA Format XXXX
i) Special Character	NA	AAA(1/8)AAA – –
j) Horizontal Space	BCE5XX	AAA* XX* AA or AAA AA HQXX
k) Vertical Space	BCD8XX	AAA AAA VQZZ
l) Ref Code	CC	AAA AAA Print "Ref" in Margin □
m) Index	CD or CE XXX CF	XXX Start Index Stop Index
n) Sub Format	AEXXXX	AAA Sub Format XXXX

X = Number: A = Text Letters

Rev. 2/15/77

ATTACHMENT 7 - CATEGORY ONE SAMPLE

1 Introduction

Content

Mission of the Service
Goals of the Internal Revenue Agent
Style of Presentation
  Content
  Introduction
  Code and Regulations
  Exercises
  Glossary
  Figures
  Names
  Tests
Authority of the Internal Revenue Agent
  Adjustments to Tax Liability
  Investigation
Summary

Mission of the Service

The mission of the Service is to encourage and achieve the highest possible degree of voluntary compliance with the tax laws and regulations and to maintain the highest degree of public confidence in the integrity and efficiency of the Service.

Goals of the Internal Revenue Agent

To do your part in accomplishing this mission, as a revenue agent you must be able to:

1. Receive, organize, and manage an inventory of tax returns to be examined.
2. Conduct field examinations of individual, partnership, fiduciary, and corporation tax returns.
3. Interact with others so that the image of the Service will be above reproach and interpersonal relationships will be conducive to the efficient conduct of official business.
4. Account for the expenditure of resources and submit other administrative reports as necessary to achieve the overall purpose of the Service.
5. Accomplish special work and projects necessary to the overall purpose of the Service.

Developing these abilities are the five goals of this training program. In this classroom segment you will be concentrating on goals 2 and 3.

You will attain goal 2 by learning:

1. Basic income tax law and concepts;
2. Techniques for examining tax returns;
3. How to write reports of your examinations; and
4. How to research the tax law.

To enable you to attain goal 3, a significant portion of this course is devoted to learning and practicing interpersonal skills which will enable you to effectively deal with people.

10) Operation and maintenance of house-trailers;
11) Tips that are incidental to any of these expenses; and
12) Other similar expenses incident to qualifying travel.

Travel expenses do not include entertainment expenses.

Business purpose. The expenses must arise from duties that require you to travel or live temporarily at some place other than your principal or regular post of duty.

You are away from home overnight if your duties require you to be away from the general area of your tax home for a period substantially longer than an ordinary day's work and, during released time while away, it is reasonable for you to need and to get sleep or rest to meet the demands of your employment or business. However, this does not mean napping in your car. You need not be away from your tax home for an entire 24-hour day or from dusk to dawn so long as your relief from duty during your absence is a sufficient period of time in which to get necessary sleep or rest.

Example 1. You are a railroad conductor. You leave your home terminal on a regularly scheduled round-trip run between two cities and return home 16 hours later. During the run you are released for 6 hours at your turnaround point where you eat two meals and rent a hotel room to get necessary sleep before starting the return trip. You are considered to be away from home overnight.

Example 2. You are a truck driver. You leave your terminal and return 11 hours later. You are released at your turnaround point for one hour in order to eat. Since you are not released to obtain necessary sleep and the brief interval of release does not constitute an adequate rest period, you are not away from home overnight.

Tax Home

Your tax home, for travel expense purposes, is your principal place of business, employment, station, or post of duty, regardless of where you maintain your family residence.

The entire city or general area in which your business or other employment is located is your tax home.

Example. You live with your family in Chicago, but work in Milwaukee, where you stay in a hotel and eat in a restaurant during the week. You return to Chicago every weekend. You may not deduct any of your expenses for travel, meals, or lodging in Milwaukee since that is your tax home and the travel over the weekends is not for a business reason.

Two places of business or employment. If you regularly work in two or more separate areas, your tax home is the general area where your ***principal*** business or employment is located.

Principal place of business or employment. The main factors in determining your principal place of business or employment are:

1) The total time ordinarily spent in performing your duties in each area;
2) The degree of your business activity in each area; and
3) The relative amount of your income from each area.

Example. You live in Cincinnati where you have a seasonal job for 8 months and earn $15,000. You work the remaining 4 months in Miami, also at a seasonal job, and earn $4,000. Cincinnati is your principal place of employment because you spend most of your time there and earn most of your income there.

Residence away from your tax home. You may be considered as traveling away from home while working in the same city in which you and your family live.

Example. Your family residence is in Pittsburgh where you work 12 weeks a year. The remainder of the time you work for the same employer in Baltimore. In Baltimore you eat in restaurants and sleep in a rooming house. Your salary is the same whether you are in Pittsburgh or Baltimore. Since you spend most of your working time and earn most of your salary in Baltimore, that city is your tax home and you may not deduct any expenses incurred for meals and lodging there. However, when you return to work in Pittsburgh you are away from home even though you stay at your family residence. Therefore, you may deduct the cost of your round trip between Baltimore and Pittsburgh, and that portion of your family living expenses for meals and lodging properly attributable to your presence in Pittsburgh while working there.

Temporary assignment. Although you regularly work or carry on your business within the city or general area of your residence, you may find it necessary or advisable, because of the demands of the job or business you are engaged in, to work or conduct business at another location from which it is not practical to return at the end of each day's work. If your work away from your regular post of duty is strictly temporary (that is, if its termination can be foreseen within a fixed and reasonably short period), your tax home does not change, and you are considered to be away from home for the entire period. ***Usually, an assignment expected to last for a year or more is not temporary.***

Return home on weekend. You may deduct the necessary travel expenses in originally getting to your temporary assignment and in returning to your tax home after the assignment is completed. Also, you may deduct reasonable expenses for meals and lodging (even for days off) while at your temporary assignment.

If you return home on your days off, you are not away from home while you are in your hometown, and the cost of your meals and lodging there is not deductible. However, you may deduct your travel expenses (including meals and lodging en route) from the area of your temporary assignment to your hometown and return if they are no more than it would have cost you for meals and lodging had you stayed at your temporary assignment. If they are more, your deduction is limited to the amount you would have spent at your temporary assignment. If you retain your hotel room while you are away, your expenses of returning home are deductible only to the extent of the amount you would have spent for meals had you stayed at your temporary assignment.

Indefinite assignment. If your job at the new location is expected to last for an indefinite period (that is, if its termination cannot be foreseen within a fixed and reasonably short period), that location becomes your new tax home and you cannot deduct your travel, meals, and lodging expenses while there. Any amounts you receive from your employer for living expenses must be included in income, even if they are designated as travel allowances and you account to your employer for them.

Whether your assignment is temporary or indefinite must be determined at the time you start work. However, a series of assignments to the same location, all for short periods but which together cover a long period, may be considered an indefinite assignment.

Probationary work period. If you accept employment that requires you to move to a new location with the understanding that you will be kept on the job if your work is satisfactory for a given test period, the employment is indefinite and you may not deduct any expenses for meals and lodging for the test period.

Transient workers. If you move from job to job, maintain no fixed residence, and are associated with no particular business locality, each place you work becomes your principal place of business and your tax home; therefore, you may not deduct your expenses for meals and lodging.

New employment. Travel expenses, including meals and lodging, incurred in seeking new employment in your present trade or business are deductible. However, these expenses are not deductible if incurred in seeking employment in a new trade or business or in seeking employment for the first time. If you are unemployed, and there is a substantial lack of continuity between the time of your past employment and the seeking of new employment, you may not deduct these expenses, even if the new employment is in the same trade or business as your last previous employment. If you are reimbursed or receive an allowance, see ***Interview expenses*** in Chapter 7.

Travel Within the United States

If your trip was entirely for business, your ordinary and necessary travel expenses may be deducted. Conversely, if your trip was solely personal, none of your travel expenses is deductible, even if you engaged in some business activity at your destination.

If your trip was primarily for business and, while at your business destination, you extend your stay for nonbusiness reasons, make a nonbusiness side trip, or engage in other nonbusi-

ATTACHMENT 8 - CATEGORY TWO SAMPLE

Retired Executives, the *Active Corps of Executives*, and all existing OEO volunteer programs. Among the latter is VISTA (Volunteers in Service to America), which has provided more than 2000 volunteers since 1965 to work on antipoverty projects in more than 1000 American communities. VISTA volunteers are at all age levels, with 8 percent over age 50. VISTA became the heart of ACTION's domestic program.

The Peace Corps. Widely heralded when it was created by President John F. Kenn[illegible] 1963, the Peace Corps was tra[illegible] TION on July 1, 1971, [illegible] its name and identity. Since [illegible]ings, almost 50,000 Americans have served as Peace Corps volunteers in more than 60 nations throughout the world. Peace Corps volunteers provided valued assistance in education, sanitation projects, agriculture, and government administration. Today, the more than 8000 Corps volunteers and trainees continue their service to nation-building as the overseas arm of ACTION.

All [illegible] OEO and other antipoverty pro[illegible]radicate the conditions that cre[illegible] poverty, but they do mark, as the [illegible]sident has said, "a commitment to pursue victory over the most ancient of mankind's enemies."

Category 2

THE CIVIL SERVICE

The National Government is the largest single employer in the United States today. It now employs some 2,800,000 persons—not counting those in uniform.[7] Of this huge number, only 537—the President, and Vice President, and the 535 members of Congress—are elected. All of the others are appointed.

It is upon the shoulders of these appointees, the federal "civil servants," that most of the burden of the day-to-day work of the National Government rests. The quality of these appointees determines in very large measure the quality of the National Government itself. No matter how wisely Congress and the President may formulate public policies, the people and the country cannot benefit unless those policies are properly and successfully executed. For, as President Woodrow Wilson once observed, the administration of governmental policy is "government in action."

DEVELOPMENT OF CIVIL SERVICE

The Constitution says very little about governmental personnel. Indeed, the only directly pertinent provision is to be found in Article II, Section 2, Clause 2 provides that the President:

> shall nominate, and, by and with the advice and consent of the Senate, shall appoint ambassadors, other public ministers, and consuls, judges of the Supreme Court, and all other officers of the United States whose appointments are not herein otherwise provided for; but the Congress may by law vest the appointment of such inferior officers, as they think proper, in the President alone, in the courts of law, or in the heads of departments.

A Good Beginning. As our first President, George Washington knew that the success or failure of the new government under the Constitution would hinge in large measure upon those whom he appointed to office. He declared his policy to be that of selecting "such persons alone . . . as shall be the best qualified." He demanded that his appointees be loyal to the Constitution; but he did not insist that they be loyal to him personally or to his political supporters.

As the party system developed, however, Washington's successors began to give weight to political considerations in the making of appointments. John Adams followed Washington's policy rather closely in that he insisted upon competence, but he also saw that none of his political enemies gained appointive office.

When Thomas Jefferson entered the White House in 1801, he found that most of the posi-

---

[7] There are some 13,000,000 civilian government employees all told, including approximately 2,800,000 now (1973) employed by the States and another 7,500,000 by their local governments.

ATTACHMENT 9 - CATEGORY THREE SAMPLE

2) You believe that you will not live long enough to benefit from recovering your annuity cost at age 65 or at your annuity starting date;

3) The election will cause the amounts you receive to be considered retirement income (provided you have reached early retirement age) on which you may be entitled to a ***credit for the elderly.***

To make the election, attach a statement to your tax return that (1) states you elect not to claim the disability income exclusion and will report your disability payments under an applicable pension or annuity rule, and (2) sets forth your qualifications for making the election. If you retired in 1976, you must also attach a certificate from a qualified physician attesting to your total and permanent disability at the time of your retirement. This election is not available to disability retirees who retired after December 31, 1975, and whose disability payments are not eligible for the disability income exclusion.

The election is applicable for the year of election and all subsequent years but may not be made for a year prior to 1976.

Military and Certain Government Disability Pensions

Effective for tax years beginning after December 31, 1975, the new law eliminates the exclusion of disability payments for personal injuries or sickness resulting from active service in the armed forces of any country or in the National Oceanic and Atmospheric Administration, the Public Health Service, or the Foreign Service by those who join these organizations after September 24, 1975.

Members of government service. If you are a member of a government service described in the preceding paragraph, you will continue to exclude such payments from gross income if:

1) On or before September 24, 1975, you were entitled to receive a disability payment;
2) On September 24, 1975, you were a member of such government service (or its reserve component), or were under a binding written commitment to become such a member;
3) You receive disability payments by reason of a combat-related injury; or
4) You would be entitled to receive disability compensation from the Veterans Administration instead, if you had filed an application to do so.

Disability benefits received from the Veterans Administration will continue to be excluded from gross income. Additionally, even if a future military retiree does not receive his disability benefits from the V.A., he can exclude an amount equivalent to the benefits he could receive from the V.A. Otherwise, future military retirees can only exclude military disability retirement payments directly attributable to combat-related injuries.

"Combat-related injury" means personal injury or sickness:

1) Which is incurred:
    a) As a direct result of armed conflict;
    b) While engaged in extra hazardous service; or
    c) Under conditions simulating war; or
2) Which is caused by an instrumentality of war.

A similar exclusion is provided for disability payments received by civilian government employees for injuries resulting from acts of terrorism and violent terrorist attacks occurring while engaged in the performance of official duties outside of the United States. However, this latter exclusion will not take effect until tax years beginning after December 31, 1976.

Other Sickness and Injury Benefits

Workmen's compensation received because of sickness or injury is fully exempt from tax. If you turn over your workmen's compensation payments to your employer, and all or part of your regular salary continues to be paid, the excess of the salary payments over the amount of workmen's compensation is taxable income.

Example. You are hurt on the job, and are out of work for 6 weeks. Your employer continues to pay your salary of $200 a week or $1,200 for the time you are absent. You receive $30 a week or a total of $180 workmen's compensation because of your injury and turn it all over to your employer under your employment contract. The $30 a week is fully excludable from your income and the balance of $170 a week is taxable income.

Other compensation. Most other amounts received as compensation for injury or illness are exempt from tax. These include:

Damages received for injury or illness;

Benefits received under an accident or health insurance policy attributable to premiums paid by you;

Disability benefits received for loss of income or earning capacity as a result of injuries under a "no fault" automobile policy;

Compensation received for permanent loss, or loss of use of a member or function of the body or for permanent disfigurement;

This compensation must be computed only in reference to the injury and not with regard to an employee's period of absence from work. These benefits are exempt from tax even though an employer finances the accident and health plan which provides these benefits.

Federal Employees' Compensation Act payments for personal injuries or sickness, including payments to beneficiaries in case of death. Additional information on the disability payment exclusion are contained in Publication 522, *Tax Information on Disability Payments.*

15. Moving Expenses

If you moved to a new residence last year because you went to work as an employee or as a self-employed individual at a new ***principal*** place of work, you may be able to deduct the reasonable expenses of your move. You may deduct your allowable moving expenses even if you do not itemize your deductions.

Subject to the limitations discussed later in this chapter, the moving expense deduction can include a number of items incurred in connection with the move. These include expenses of:

1) Travel to the new location;
2) Movement of household goods and personal effects;
3) Premove househunting trips;
4) Temporary quarters at the new location; and
5) Disposition of an old residence and acquisition of a new one.

If you moved to a new residence outside the country, see Publication 54, *Tax Guide for U.S. Citizens Abroad,* for special rules affecting the deductibility of moving expenses.

How to Qualify for the Deduction

Your moving expenses are deductible provided you meet the following requirements.

1) Distance: Your new principal place of work must be at least 50 miles farther from your old residence than your old residence was from your old place of work. If you had no old place of work, the new place of work must be at least 50 miles from your old residence. If the distance is not 50 miles or more, your moving expenses are not deductible.

2) Time: During the 12-month period immediately following the move, you must be emcludable from income. See an exception later under *Qualified Second Household.*

Two qualified spouses. If both you and your spouse qualify for the deduction for excess foreign living costs, you may each be entitled to separate housing expense amounts if you maintain separate households. See the discussion of *Separate households,* later under *Married Couples Abroad.*

Income received in a later year. If you receive income in the current year that was earned during any of the three prior years, you may have to include an amount as an excess housing deduction in your income in the year the income is received.

Excess housing deduction. To find if you have an excess housing deduction to include in income, recompute the amount of the allowable housing expenses for both the year you performed the services that earned the income and the year you received the income. You do this by including the income received in the later year in the year it was earned, not in the year it was received.

The amount of the excess housing deduction for a tax year is any excess of:

1) The total amount allowed as a housing deduction for that tax year and all prior tax years, over
2) The total amount that would have been allowed as a housing deduction for that tax year and all prior tax years by including the income received in the later year as income received in the year it was earned.

If an amount was required to be included in income as an excess housing deduction in a prior year, the amount of the excess housing deduction for the current year is reduced by the amount that was previously included in income.

The excess housing deduction is included in your gross income in the year you receive the previously earned income.

If your total deduction for excess foreign living costs in any year for which you are required to recompute your housing deduction exceeds the limit on the amount of the deduction (described later) for that year, the amount of your housing deduction for that year is equal to the excess of net foreign source earned income over the total of the cost-of-living differential, schooling expenses, home leave transportation expenses, and hardship amount.

Example 1. You are a U.S. citizen and first claim the deduction for excess foreign living costs in 1979. In 1980 you receive a $10,000 bonus for services performed in 1979. In 1980 you cannot deduct housing expenses because you resided in the United States in that year. Your 1979 housing income was $20,000 and the total of your allowable cost-of-living differential and schooling and housing expenses was $9,500, of which $5,000 was the amount of your allowable housing expenses. Your excess housing deduction is figured as follows:

	Actual	Recomputed
1979 housing income	$20,000	$30,000
Minus cost-of-living differential and schooling and housing expenses	9,500	9,500
	$10,500	$20,500
	20%	20%
1979 base housing amount	$ 2,100	$ 4,100
Housing expenses	$ 5,000	$ 5,000
Less base housing amount	2,100	4,100
1979 housing deduction	$ 2,900	$ 900

Your excess housing deduction is $2,000, the excess of $2,900 (the actual 1980 housing deduction, 0, plus prior year housing deduction, $2,900) over $900 (the recomputed amounts for 1980, 0, plus the recomputed amount for the prior year, $900). The $2,000 must be included in your 1980 gross income in addition to your $10,000 bonus. No amendment to your 1979 return is required.

Example 2. You are a U.S. citizen and claim the deduction for excess foreign living costs in 1978, 1979, and 1980. During 1980 you received deferred income in the amount of $40,000, one-half of which was earned in each of the years 1978 and 1979. In 1980 your housing expenses are not deductible because they are not more than your base housing amount. Your actual income and expenses are as follows:

	1978	1979	1980
Housing income	$30,000	$32,000	$74,000
Housing expenses	$ 6,800	$ 6,800	$ 7,000
Schooling and home leave transportation expenses	$ 6,200	$ 6,600	$ 6,600

Your excess housing deduction is figured as follows:

	Actual			Recomputed		
	1978	1979	1980	1978	1979	1980
Housing income	$30,000	$32,000	$74,000	$50,000	$52,000	$34,000
Minus schooling, home leave transportation, and housing expenses	13,000	13,400	13,600	13,000	13,400	13,600
	$17,000	$18,600	$60,400	$37,000	$38,600	$20,400
	20%	20%	20%	20%	20%	20%
Base housing amount	$ 3,400	$ 3,720	$12,080	$ 7,400	$ 7,720	$ 4,080
Housing expenses	$ 6,800	$ 6,800	$ 7,000	$ 6,800	$ 6,800	$ 7,000
Base housing amount	3,400	3,720	12,080	7,400	7,720	4,080
Housing deduction	$ 3,400	$ 3,080	$ -0-	$ -0-	$ -0-	$ 2,920

The amount of your 1980 excess housing deduction is $3,560, the excess of the total actual housing deductions ($6,480) over the recomputed housing deductions ($2,920). The $3,560 must be added to the 1980 income reported on your return, and no adjustments are required to your 1978 or 1979 returns.

Qualified Second Household

Special rules apply to the computation of your excess foreign living costs if you maintain a household in a foreign country for your spouse or your dependents at a place other than your tax home, because the living conditions at your tax home are dangerous, unhealthful, or otherwise adverse. This second household is a qualified second household. You may have only one qualified second household at any time.

If your tax home is in a hardship area (defined earlier), living conditions are considered to be adverse. However, your tax home does not have to be in a hardship area for you to have a qualified second household. Adverse living conditions include a state of warfare or civil insurrection in the general area of your tax home, or, if your abode is on the business premises of your employer and, because of the nature of the business premises (for example, a construction site or drilling rig), it is not feasible to provide family housing.

If both you and your spouse qualify for the deduction for excess foreign living costs, special rules apply for a qualified second household. See the discussion later under *Married Couples Abroad.*

If you maintain a qualified second household, the general rules just discussed are modified as follows.

Cost-of-living differential. The cost-of-living differential for the part of the tax year that you maintain a qualified second household is based on the amount specified for the location of your second household, not your tax home. You receive no separate cost-of-living differential for your tax home during that period.

The cost-of-living differential is not denied for days that you exclude from income the value of meals and lodging furnished for the convenience of your employer. When finding the differential on the cost-of-living table, you are included as a family member, except for the days that you exclude from income the value of both meals and lodging, even if you were not actually present in the qualified second household.

Housing expenses. If you maintained a qualified second household, you may deduct hous-

ATTACHMENT 10 - CATEGORY FOUR SAMPLE

Table 1.5—Joint Returns: Sources of Income and Adjustments by Size of Adjusted Gross Income

[All figures are estimates based on samples—money amounts are in thousands of dollars]

Size of adjusted gross income	Number of returns	Adjusted gross income less deficit	Salaries and wages		Interest received		Alimony received		State income tax refunds		Pensions and annuities			
			Number of returns	Amount	Number of returns	Amount	Number of returns	Amount	Number of returns	Amount	Total		In adjusted gross income	
											Number of returns	Amount	Number of returns	Amount
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)
All returns, total	**44,483,348**	**947,136,976**	**39,596,535**	**790,116,918**	**28,452,437**	**40,162,604**	**9,299**	**41,902**	**7,382,644**	**2,035,521**	**4,453,103**	**25,556,721**	**4,238,162**	**23,508,769**
Under $2,000	927,634	−5,245,498	493,262	1,985,407	430,776	664,286	*21	*85	36,107	17,530	84,556	345,296	70,136	188,788
$2,000 under $4,000	1,212,049	3,786,007	854,743	2,747,791	526,631	510,075	—	—	18,314	2,767	177,891	557,873	157,296	374,931
$4,000 under $6,000	1,851,164	9,442,482	1,342,718	6,428,276	811,063	1,075,442	—	—	35,417	6,196	415,479	1,320,581	397,862	1,146,584
$6,000 under $8,000	2,444,649	17,173,381	1,857,342	11,972,461	1,242,528	1,944,569	—	—	67,101	12,380	483,703	1,853,036	471,049	1,747,432
$8,000 under $10,000	2,544,225	22,856,362	2,065,420	17,301,148	1,281,131	2,023,590	—	—	126,065	36,663	478,513	2,095,203	449,804	1,970,113
$10,000 under $12,000	2,848,787	31,366,447	2,460,814	25,364,027	1,322,126	1,879,927	*603	*559	170,742	30,592	372,981	2,048,736	360,018	1,908,852
$12,000 under $14,000	3,030,075	39,411,081	2,724,712	33,366,176	1,568,070	1,870,322	—	—	225,975	46,244	328,177	1,911,106	317,825	1,806,433
$14,000 under $16,000	3,154,980	47,351,850	2,913,545	41,720,820	1,668,290	1,707,191	*30	*18	316,697	59,655	274,262	1,796,523	256,439	1,603,262
$16,000 under $18,000	3,290,400	55,978,937	3,080,137	50,044,918	1,882,350	1,814,613	*2,312	*8,355	479,623	100,653	262,680	1,619,956	248,643	1,489,537
$18,000 under $20,000	3,275,666	62,238,619	3,100,404	56,279,416	1,990,033	1,888,325	*176	*162	584,303	114,508	234,413	1,464,378	214,824	1,355,881
$20,000 under $25,000	7,223,411	161,656,254	6,925,249	147,776,409	4,979,091	4,286,867	*310	*357	1,715,064	381,530	456,927	2,920,577	440,141	2,746,838
$25,000 under $30,000	4,895,167	133,513,218	4,663,726	120,033,635	3,846,945	3,843,080	*2,130	*514	1,367,228	311,347	311,116	2,317,649	299,440	2,208,378
$30,000 under $50,000	6,117,176	223,641,677	5,700,763	188,569,321	5,315,953	8,412,327	2,545	24,683	1,824,221	562,520	438,794	3,839,274	425,691	3,600,843
$50,000 under $100,000	1,346,853	88,049,930	1,146,497	58,081,818	1,276,272	4,890,585	971	3,612	343,396	226,993	108,271	1,133,553	104,229	1,055,571
$100,000 under $200,000	260,728	34,254,617	216,793	19,780,041	252,065	2,016,074	121	622	59,324	81,391	19,574	236,461	19,150	217,627
$200,000 under $500,000	53,192	14,807,377	44,314	6,908,999	52,041	916,165	64	2,069	11,426	31,929	4,936	78,154	4,804	71,641
$500,000 under $1,000,000	5,560	3,690,712	4,698	1,180,738	5,466	232,383	**16	**867	1,261	8,015	622	12,991	611	11,735
$1,000,000 or more	1,632	3,163,523	1,398	575,518	1,606	186,783	**	**	380	4,609	208	5,375	200	4,325
Taxable returns, total	**38,587,496**	**918,254,754**	**35,540,678**	**786,751,277**	**25,740,626**	**36,394,161**	**9,241**	**41,767**	**7,128,036**	**1,953,493**	**3,522,983**	**22,240,313**	**3,367,980**	**20,791,339**
Under $2,000	14,399	−527,432	5,268	174,818	10,520	115,670	—	—	1,195	1,651	120	905	120	891
$2,000 under $4,000	2,273	7,979	1,151	15,453	2,131	5,591	—	—	*35	*96	*31	*222	*31	*221
$4,000 under $6,000	222,468	1,246,800	184,459	966,320	85,612	114,203	—	—	*357	*86	29,624	114,470	29,596	112,217
$6,000 under $8,000	1,278,156	9,212,442	1,017,769	6,703,563	723,207	1,064,489	—	—	29,408	4,178	266,775	976,348	258,427	930,737
$8,000 under $10,000	2,164,300	19,473,847	1,765,696	14,726,858	1,117,845	1,783,699	—	—	77,822	18,009	441,354	1,927,493	415,654	1,812,300
$10,000 under $12,000	2,619,048	28,866,223	2,285,483	23,570,445	1,210,594	1,693,504	*566	*509	141,301	23,935	357,624	1,942,878	344,669	1,819,259
$12,000 under $14,000	2,944,697	38,309,093	2,669,935	32,717,578	1,513,482	1,761,781	—	—	215,234	42,072	323,701	1,901,256	313,349	1,796,981
$14,000 under $16,000	3,091,031	46,407,332	2,867,726	41,204,199	1,630,627	1,613,065	*30	*18	309,675	57,746	271,123	1,780,331	253,300	1,587,818
$16,000 under $18,000	3,241,183	55,155,260	3,045,526	49,583,134	1,853,843	1,770,264	*2,312	*8,355	469,650	96,748	262,344	1,617,208	248,307	1,486,789
$18,000 under $20,000	3,239,117	61,548,398	3,077,676	55,951,392	1,959,276	1,835,599	*176	*162	580,067	112,113	232,253	1,445,931	212,664	1,337,610
$20,000 under $25,000	7,169,364	160,460,052	6,888,220	147,211,923	4,945,330	4,241,877	*310	*357	1,709,703	377,891	456,153	2,916,631	439,367	2,743,109
$25,000 under $30,000	4,864,369	132,680,883	4,646,306	119,717,451	3,821,218	3,813,167	*2,130	*514	1,362,383	309,719	310,144	2,316,247	298,468	2,207,047
$30,000 under $50,000	6,080,682	222,272,314	5,679,745	188,081,352	5,288,526	8,371,313	2,545	24,683	1,817,071	557,512	438,520	3,836,187	425,417	3,597,756
$50,000 under $100,000	1,336,373	87,378,587	1,139,271	57,749,358	1,268,029	4,863,903	971	3,612	341,807	225,948	107,918	1,132,229	103,887	1,054,280
$100,000 under $200,000	259,734	34,126,267	216,102	19,723,485	251,339	2,011,789	121	622	59,274	81,289	19,535	235,497	19,111	216,663
$200,000 under $500,000	53,119	14,789,104	44,257	6,901,325	51,981	915,383	64	2,069	11,415	31,877	4,934	78,114	4,802	71,603
$500,000 or more	7,183	6,847,606	6,088	1,752,624	7,066	418,863	16	867	1,639	12,620	830	18,366	811	16,060
Total nontaxable returns	**5,895,852**	**28,882,222**	**4,055,857**	**23,365,642**	**2,711,811**	**3,768,443**	***58**	***135**	**254,608**	**82,027**	**930,120**	**3,316,408**	**870,182**	**2,717,430**
All returns, summary:														
Under $5,000	2,929,584	2,098,657	1,921,957	7,188,641	1,269,676	1,511,742	*21	*85	60,881	21,784	401,620	1,376,848	356,088	955,869
$5,000 under $10,000	6,050,137	45,916,076	4,691,528	33,246,442	3,022,453	4,706,220	—	—	222,123	53,752	1,238,522	4,795,140	1,190,079	4,471,978
$10,000 under $15,000	7,443,907	93,492,436	6,619,703	78,642,332	3,682,688	4,648,269	*603	*559	532,242	97,683	851,469	4,889,798	815,579	4,514,332
$15,000 under $20,000	8,156,001	142,854,499	7,659,909	128,133,025	4,748,181	4,512,110	*2,518	*8,534	1,245,098	253,969	621,044	3,950,900	582,170	3,649,633
$20,000 or more	19,903,719	662,777,308	18,703,438	542,906,478	15,729,439	24,784,263	6,157	32,723	5,322,300	1,608,334	1,340,448	10,544,034	1,294,266	9,916,958

Footnotes at end of table.

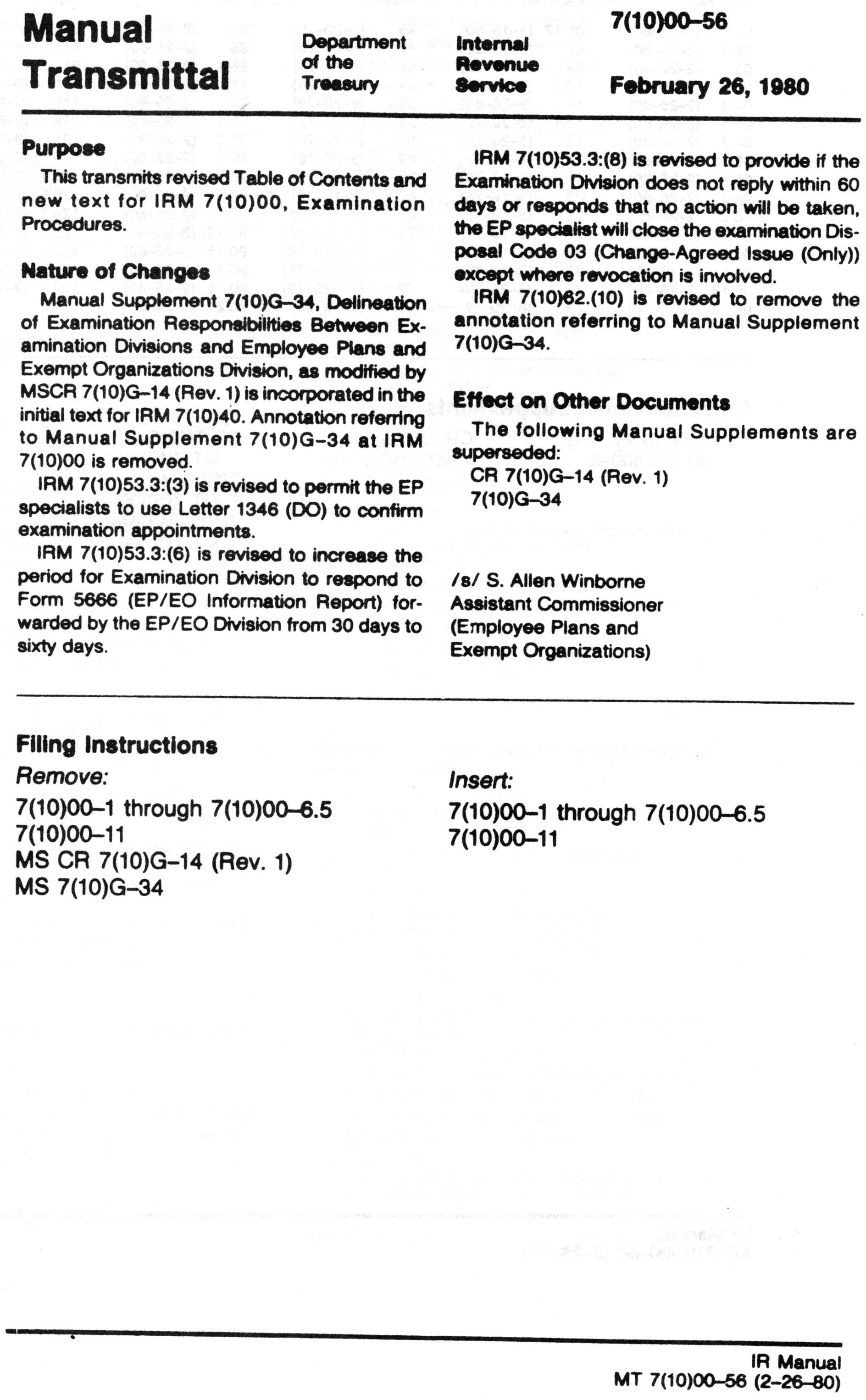
Table 1.6—Returns With Income Earned Abroad: Selected Items by Size of Adjusted Gross Income

[All figures are estimates based on samples - money amounts are in thousands of dollars]

Size of adjusted gross income	Income earned abroad		Amount exempted, excluded, or deducted		Salaries and wages		Total net profit less loss from business activities[1]	
	Number of returns	Amount	Number of returns	Amount	Number of returns	Amount	Number of returns	Amount
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
All returns with income earned abroad								
Total	**169,951**	**5,772,795**	**166,801**	**2,367,749**	**134,749**	**4,214,735**	**17,483**	**11,220**
No adjusted gross income	21.443	296.297	*21.196	*279.969	*4.277	*7.110	*308	*3.639
$1 under $10,000	54.599	944 707	54.599	733.380	39.624	270.156	*6.604	*−14.692
$10.000 under $20.000	26.392	761.146	26.206	428 866	25.317	438.999	*2.091	*−8.304
$20,000 under $30,000	21.435	714.762	20.012	262.310	19.862	505.104	*1.819	*1.986
$30.000 under $50,000	22.093	1.040.476	21,210	293.057	21.946	969.378	2.597	7.981
$50,000 under $100,000	19.220	1.403.379	18.952	295.286	19.113	1.377.564	3.109	−9.558
$100.000 under $200,000	4.257	498,278	4.147	67.531	4.112	533.861	742	15.122
$200,000 under $500,000	464	96.414	432	6.664	457	98.832	188	9.842
$500.000 under $1,000.000	40	12 609	39	594	36	10.285	20	3.378
$1.000.000 or more	8	4.729	8	93	5	3.446	5	1.825
Returns with tax-exempt foreign income								
Total	**135,790**	**3,759,102**	**135,790**	**1,955,180**	**101,617**	**2,168,942**	**10,543**	**20,929**
No adjusted gross income	*21.147	*287.774	*21.147	*278.325	*4.228	*3.626	*12	*−527
$1 under $10,000	51.173	857.110	51.173	688.989	36.771	226.950	*3.178	*162
$10.000 under $20.000	23.776	698.814	23.776	395.983	22.750	362.080	*1.856	*−11.164
$20.000 under $30,000	16.649	556,755	16.649	235.316	15,076	365.112	*1.623	*2.693
$30.000 under $50.000	13.174	582.550	13.174	185.024	13.076	512.945	1.888	5.923
$50.000 under $100.000	8.374	581.097	8.374	144.075	8.325	519.862	1.611	5.382
$100.000 under $200.000	1.277	145.005	1.277	23.550	1.181	133.100	266	10.874
$200.000 under $500.000	195	41.356	195	3.420	191	39.559	93	5.327
$500,000 under $1.000.000	20	6.830	20	422	16	3.971	13	2.177
$1,000,000 or more	5	1.811	5	78	3	1.736	3	83
Returns with exclusion of income earned in hardship areas								
Total	**6.136**	**272.267**	**6.136**	**98.100**	**6.126**	**178.337**	**2.966**	**−16.070**
No adjusted gross income	—	—	—	—	—	—	—	—
$1 under $10.000	*2.853	*86.001	*2 853	*42 795	*2.853	*43 206	*2.853	*−16 214
$10.000 under $20.000	—	—	—	—	—	—	—	—
$20 000 under $30.000	*981	*45 994	*981	*16 223	*981	*30 199	*98	*−14
$30 000 under $50.000	*1 054	*55.479	*1 054	*20 020	*1 054	*35 774	—	—
$50 000 under $100.000	*1 238	*83 530	*1 238	*18 585	*1 228	*67 386	*10	*157
$100 000 under $200.000	*5	*428	*5	*400	*5	*1 014	—	—
$200 000 under $500.000	*5	*835	*5	*78	*5	*757	*5	*1
$500 000 under $1 000 000	—	—	—	—	—	—	—	—
$1 000 000 or more	—	—	—	—	—	—	—	—
Returns with deduction for expense of living abroad								
Total	**28.644**	**1,751.754**	**28.644**	**314.468**	**27.872**	**1,832.351**	**3.372**	**346**
No adjusted gross income	*49	*3 485	*49	*1 643	*49	*3 485	*49	*−873
$1 under $10.000	*573	*1.596	*573	*1 596	—	—	*573	*1 360
$10 000 under $20 000	*5 608	*144 437	*5 608	*32 883	*5 559	*102 910	*49	*2 587
$20 000 under $30 000	*2.382	*65 157	*2 382	*10 771	*2 382	*74 564	—	—
$30 000 under $50 000	6 982	369 873	6 982	88 014	6 933	388 085	709	2 058
$50 000 under $100 000	9 870	756 038	9 870	*32 626	9 822	801 440	1 439	−15 917
$100 000 under $200 000	2 901	350 921	2 901	43 581	2 852	397 979	438	4 387
$200 000 under $500 000	257	52 214	257	3 167	254	56 526	106	3 800
$500 000 under $1 000 000	19	5 115	19	172	**21	**7 361	**9	**2 943
$1 000 000 or more	3	2 918	3	16	**	**	**	**

Footnotes at end of table

ATTACHMENT 11 - CATEGORY FIVE SAMPLE



Manual Transmittal

Department of the Treasury

Internal Revenue Service

7(10)00-56

February 26, 1980

Purpose

This transmits revised Table of Contents and new text for IRM 7(10)00, Examination Procedures.

Nature of Changes

Manual Supplement 7(10)G-34, Delineation of Examination Responsibilities Between Examination Divisions and Employee Plans and Exempt Organizations Division, as modified by MSCR 7(10)G-14 (Rev. 1) is incorporated in the initial text for IRM 7(10)40. Annotation referring to Manual Supplement 7(10)G-34 at IRM 7(10)00 is removed.

IRM 7(10)53.3:(3) is revised to permit the EP specialists to use Letter 1346 (DO) to confirm examination appointments.

IRM 7(10)53.3:(6) is revised to increase the period for Examination Division to respond to Form 5666 (EP/EO Information Report) forwarded by the EP/EO Division from 30 days to sixty days.

IRM 7(10)53.3:(8) is revised to provide if the Examination Division does not reply within 60 days or responds that no action will be taken, the EP specialist will close the examination Disposal Code 03 (Change-Agreed Issue (Only)) except where revocation is involved.

IRM 7(10)62.(10) is revised to remove the annotation referring to Manual Supplement 7(10)G-34.

Effect on Other Documents

The following Manual Supplements are superseded:

CR 7(10)G-14 (Rev. 1)
7(10)G-34

/s/ S. Allen Winborne
Assistant Commissioner
(Employee Plans and
Exempt Organizations)

Filing Instructions

Remove:	*Insert:*
7(10)00-1 through 7(10)00-6.5	7(10)00-1 through 7(10)00-6.5
7(10)00-11	7(10)00-11
MS CR 7(10)G-14 (Rev. 1)	
MS 7(10)G-34	

IR Manual
MT 7(10)00-56 (2-26-80)

List of Current Pages

Page No.		Page No.		Page No.		Page No.		Page No.	
01	(2-26-80)	06.17	(1-18-80)	45	(8-20-79)	83	(2-25-80)	90.23	(2-25-80)
02.1	(2-26-80)	07	(2-15-80)	47	(8-20-79)	85	(2-25-80)	90.25	(9-19-79)
03	(2-26-80)	09	(7-12-79)	49	(8-20-79)	86.1	(2-25-80)	91	(9-19-79)
04.1	(2-26-80)	10.1	(7-12-79)	51	(8-20-79)	87	(2-25-80)	111	(9-17-79)
04.3	(2-26-80)	11	(2-26-80)	53	(8-20-79)	89	(2-25-80)	113	(9-17-79)
04.5	(2-26-80)	12.1	(8-20-79)	55	(8-20-79)	90.1	(2-25-80)	119	(3-9-79)
04.7	(2-26-80)	13	(8-20-79)	61	(8-20-79)	90.3	(2-25-80)	121	(3-19-79)
05	(2-26-80)	15	(3-9-79)	63	(8-20-79)	90.5	(2-25-80)	127	(2-14-80)
06.1	(2-26-80)	17	(9-17-79)	65	(8-20-79)	90.7	(2-25-80)	128.1	(2-14-80)
06.3	(2-26-80)	23	(5-19-78)	67	(2-15-80)	90.9	(2-25-80)	128.3	(2-14-80)
06.5	(2-26-80)	25	(5-19-78)	69	(2-15-80)	90.11	(2-25-80)	128.5	(2-14-80)
06.7	(1-18-80)	27	(5-19-78)	73	(12-29-78)	90.13	(2-25-80)	128.7	(2-14-80)
06.9	(1-18-80)	29	(5-19-78)	75	(12-29-78)	90.15	(2-25-80)	129	(3-19-79)
06.11	(1-18-80)	31	(8-20-79)	77	(12-29-78)	90.17	(2-25-80)	131	(3-19-79)
06.13	(1-18-80)	37	(8-20-79)	79	(12-29-78)	90.19	(2-25-80)	137	(3-19-79)
06.15	(1-18-80)	39	(8-20-79)	81	(9-19-79)	90.21	(2-25-80)		

Current Manual Supplements

7(10)G-10 (Rev. 1)
MT 7(10)00-5

CR 7(10)G-28
MT 7(10)00-14

CR 7(10)G-32
MT 7(10)00-21
7(10)G-42
MT 7(10)00-45
7(10)G-43 (Rev. 1)
MT 7(10)00-51

Table of Contents

7(10)10
(Reserved)

7(10)20
General Guidelines for EP and EO Examinations
- Awards of Attorney's Fees in Tax Cases 7(10)21
- Summons 7(10)22
  - Statutory Provisions 7(10)22.1
  - Summons to Appear to Testify, Produce Books, etc. 7(10)22.2
  - Forwarding Copies of Summons Enforcement Requests on Churches to the National Office 7(10)22.3
- Federal Grand Juries 7(10)23

7(10)30
Instructions to Examiners
- Treasury Enforcement Communications System (TECS) 7(10)31
- Procedures for Use of TECS 7(10)32
- Requesting Engineering Assistance 7(10)33

7(10)40
Delineation of Examination Responsibilities Between District Employee Plans and Exempt Organizations Division and Examination Division
- General 7(10)41
- Procedures 7(10)42
- Examination Responsibilities 7(10)43
  - Organizations Exempt under IRC 501(a) (other than one described in section 401(a)) and Farmers' Cooperatives Exempt under IRC 521 7(10)43.1
  - Examination of Forms 1120-POL, Forms 1120-H, and Forms 1120 Filed by Taxable Private Foundations 7(10)43.2
  - Group Legal Service Plans and Trusts (IRC 120 and IRC 501(c)(20)) 7(10)43.3
  - Examination of Religious and Apostolic Organizations Described in IRC-501(d) 7(10)43.4
  - Non-Exempt Charitable Trusts and Split-Interest Trusts Described in IRC 4947 7(10)43.5
- Employee Plans Referrals from Examination Division 7(10)44
- Employee Plans Initiated Examinations 7(10)45
- Collateral Examinations 7(10)46
- Statute Controls on Returns 7(10)47

7(10)50
Employee Plans Enforcement Program
- General 7(10)51
  - Scope 7(10)51.1
  - Control of Cases 7(10)51.2
  - Survey Before and After Assignment 7(10)51.3
  - Fraud and Information Referrals 7(10)51.4
- Examination Referrals 7(10)52
  - Objectives 7(10)52.1
  - General 7(10)52.2
  - Screening and Controlling Audit Referrals 7(10)52.3
  - Examination of Referrals 7(10)52.4
  - No Change Cases 7(10)52.5
  - Change cases 7(10)52.6
- Independent Examinations 7(10)53
  - Objectives 7(10)53.1
  - General 7(10)53.2
  - General Examination Procedures 7(10)53.3
  - Reports and Procedures—Agreed Cases 7(10)53.4
  - Reports and Procedures—Unagreed Cases 7(10)53.5
- Employee Plans Audit Guidelines Handbook 7(10)54
- Form 5330 Examinations 7(10)55
  - General 7(10)55.1
  - Pre-1975 Prohibited Transactions 7(10)55.2
  - Reports and Procedures—Agreed Cases 7(10)55.3
  - Reports and Procedures—Unagreed Cases 7(10)55.4
  - Request for Waivers of Exemption from IRC 4971/4975 7(10)55.5
  - Request for Waiver of the 100 Percent Tax 7(10)55.6
  - Claims 7(10)55.7
- Special Projects 7(10)56
  - Survey of Individual Retirement Account (IRA) Sponsors and Processing of IRA Complaints 7(10)56.1
    - Survey Procedures 7(10)56.11
    - IRA Complaint Procedures 7(10)56.12
    - Closing Procedures 7(10)56.13
    - Reporting Requirements 7(10)56.14
    - Penalties 7(10)56.15

7(10)60
Exempt Organizations Examination Program
- General 7(10)61
  - Introduction 7(10)61.1
  - National Office Review of Certain Cases 7(10)61.2
- General Instructions to Exempt Organization Examiners 7(10)62
  - Introduction 7(10)62.1
  - Coordination of Examination of Exempt Organization Returns 7(10)62.2
  - Private Foundation Cases 7(10)62.3
    - Introduction 7(10)62.31
    - Scope 7(10)62.32
  - Forwarding IRC 501(c)(9) Cases to the National Office When Revocation Is Contemplated 7(10)62.4
  - Extensions to the Period of Limitation 7(10)62.5
  - Procedures for Agreed Cases 7(10)62.6
  - Procedures for Unagreed Cases 7(10)62.7
  - Advance Approval of Proposed Correction 7(10)62.8
  - Examinations of Central and Intermediate or Subordinate Organizations 7(10)62.9

[Supplemented by MS CR 7(10)G-28, and 7(10)G-42]

7(10)10 *(1-28-77)*
(Reserved)

7(10)20 *(3-19-79)*
General Guidelines for EP and EO Examinations

7(10)21 *(3-19-79)*
Awards of Attorney's Fees in Tax Cases

(1) P.L. 94-559 (October 19, 1976), 90 Stat. 2641, provides courts discretionary authority to award attorney's fees to prevailing parties, other than the United States in, ". . . any civil action or proceeding by or on behalf of the United States of America to enforce, or charging a violation of, a provision of the United States Internal Revenue Code. . . ."

(2) The scope of this provision has not yet been fully determined; but, it is clear that in certain cases attorney's fees may be awarded when a taxpayer is able to prove harassment or bad faith on the part of the Internal Revenue Service. The Act appears to be applicable to a wide variety of litigation in the General Litigation function, including, but not limited to, suits to reduce a tax claim to judgment, to enforce a levy, to foreclose a tax lien, to recover an erroneous refund, to set aside a fraudulent conveyance or establish transferee liability, to open a safe deposit box and to enforce summonses seeking information in aid of an investigation. Additionally, the Act would seem pertinent to those occasions where the United States institutes an action seeking injunctive relief to enforce a provision of the Internal Revenue Code. This would occur, for instance, in an injunction action against a tax return preparer under *Code* §7407, as amended by section 1203 of the Tax Reform Act of 1976. The Act does not apply to United States Tax Court cases or to the ordinary refund suit since the taxpayer is the petitioner or plaintiff in these cases.

(3) The principles Service employees are to follow in carrying out their duties are set forth in Rev. Proc. 64-22, 1964-1 C.B. (Part I) 689.

The pertinent part of Rev. Proc. 64-22 follows:

> ". . . It is the responsibility of each person in the Service, charged with the duty of interpreting the law, to try to find the true meaning of the statutory provision and not adopt a strained construction in the belief that he is "protecting the revenue." The revenue is properly protected only when we ascertain and apply the true meaning of the statute.
>
> "The Service also has the responsibility of applying and administering the law in a reasonable, practical manner. Issues should only be raised by examining officers when they have merit, never arbitrarily or for trading purposes. At the same time, the examining officer should never hesitate to raise a meritorious issue. It is also important that care be exercised not to raise an issue or to ask a court to adopt a position inconsistent with established Service position.
>
> "Administration should be both reasonable and vigorous. It should be conducted with as little delay as possible and with great courtesy and consideration. It should never try to overreach and should be reasonable within the bounds of law and sound administration. . . ."

(4) All EP/EO specialists will strictly adhere to the principles set forth in (3) above so as to avoid the application of P.L. 94-559. The principles in Rev. Proc. 64-22 are printed on page 2 of each Internal Revenue Bulletin and are easily accessible to persons within and outside the Service and Office of the Chief Counsel.

(5) The impact of P.L. 94-559 should be considered any time a collection suit is being recommended. Any questions arising on this issue should be cleared through District Counsel.

(6) The United States Tax Court has determined that it is not authorized by the provisions of P.L. 94-559 or any other provision of law to make any allowance of attorney fees to a petitioner. *Key Buick Company v. Commissioner*, 68 T.C. No. 17 (May 16, 1977).

7(10)22 *(5-1-79)*
Summons

7(10)22.1 *(5-1-79)*
Statutory Provisions

(1) The Code provisions relating to the authority to issue and enforce a summons are as follows:

(a) IRC 7602-Examination of Books and Witnesses; also IRC 6420 and 6421.

(b) IRC 7603-Service of Summons.

(c) IRC 7604-Enforcement of Summons.

(d) IRC 7605-Time and Place of Examination.

(e) IRC 7402-Jurisdiction of District Courts.

(f) IRC 7210-Failure to Obey Summons.

(2) Under IRC 7602, the Secretary or the Secretary's delegate is authorized to examine any books, papers, records or other data and to

take such testimony, under oath, as may be relevant or material to such inquiry.

(3) Under IRC 7604(a) and 7402(b), the district courts of the United States have jurisdiction to compel attendance of any person summoned and to issue orders, processes, and judgements as may be necessary or appropriate for the enforcement of the internal revenue laws.

(4) If the testimony of any particular person is important and necessary to disclose facts to aid in determining the correct amount of tax, and such person refuses to give testimony, or to permit an inspection of records, or property, etc., his/her attention should be called to the appropriate sections of the law. If the person still refuses to grant the examiner's request, an immediate report should be made to the District Director for decision as to whether the matter should be placed in the hands of District Counsel for enforcement.

7(10)22.2 *(7-23-79)*
Summons to Appear to Testify, Produce Books, etc.

(1) The provisions of IRM 4022, Summons to Appear to Testify, Produce Books, etc., will be followed by EP/EO personnel. Delegation Order No. 4, as revised, identifies those IRS officers and employees authorized to issue summonses. The summons reporting provisions in IRM 4022 for Form 5535 (Administrative Summons Control Form) will be followed by EP/EO personnel. Any reference to the Examination function or to revenue agent will also apply to the EP/EO Division or to employee plans and exempt organization specialists. Any reference to the Returns Program Manager will apply to the Chief, Technical Staff.

(2) Exempt Organization specialists will not issue a summons for the sole purpose described in Regulation 301.7605-1(c)(2)(iii), i.e., to obtain information for the purpose of ascertaining or verifying payments made by the organization to another person in determining the tax liability of the recipient, such as payments of salaries, wages, or other forms of compensation.

7(10)22.3 *(7-23-79)*
Forwarding Copies of Summons Enforcement Requests on Churches to the National Office

(1) Key districts will forward two copies of each request to District Counsel for summons enforcement to the regional office immediately upon forwarding such a request to District Counsel from the EP/EO Division.

(2) The regional office will maintain a copy of the request as an advisory document on summons activity within the region. The other copy will be forwarded to the National Office, Exempt Organizations Division (Attention: E:EO:O:P) immediately upon receipt from the key district.

(3) This procedure applies only to exempt organization matters and is not applicable to summons enforcement action involving employee plans.

7(10)23 *(5-1-79)*
Federal Grand Juries

The provisions of IRM 4500 relating to Federal grand jury information and activities will be followed by EP/EO personnel. Any reference to the Examination function or to revenue agent will also apply to EP/EO Division or to employee plans and exempt organization specialists.

7(10)30 *(4-19-79)*
Instructions to Examiners

7(10)31 *(4-19-79)*
Treasury Enforcement Communications System (TECS)

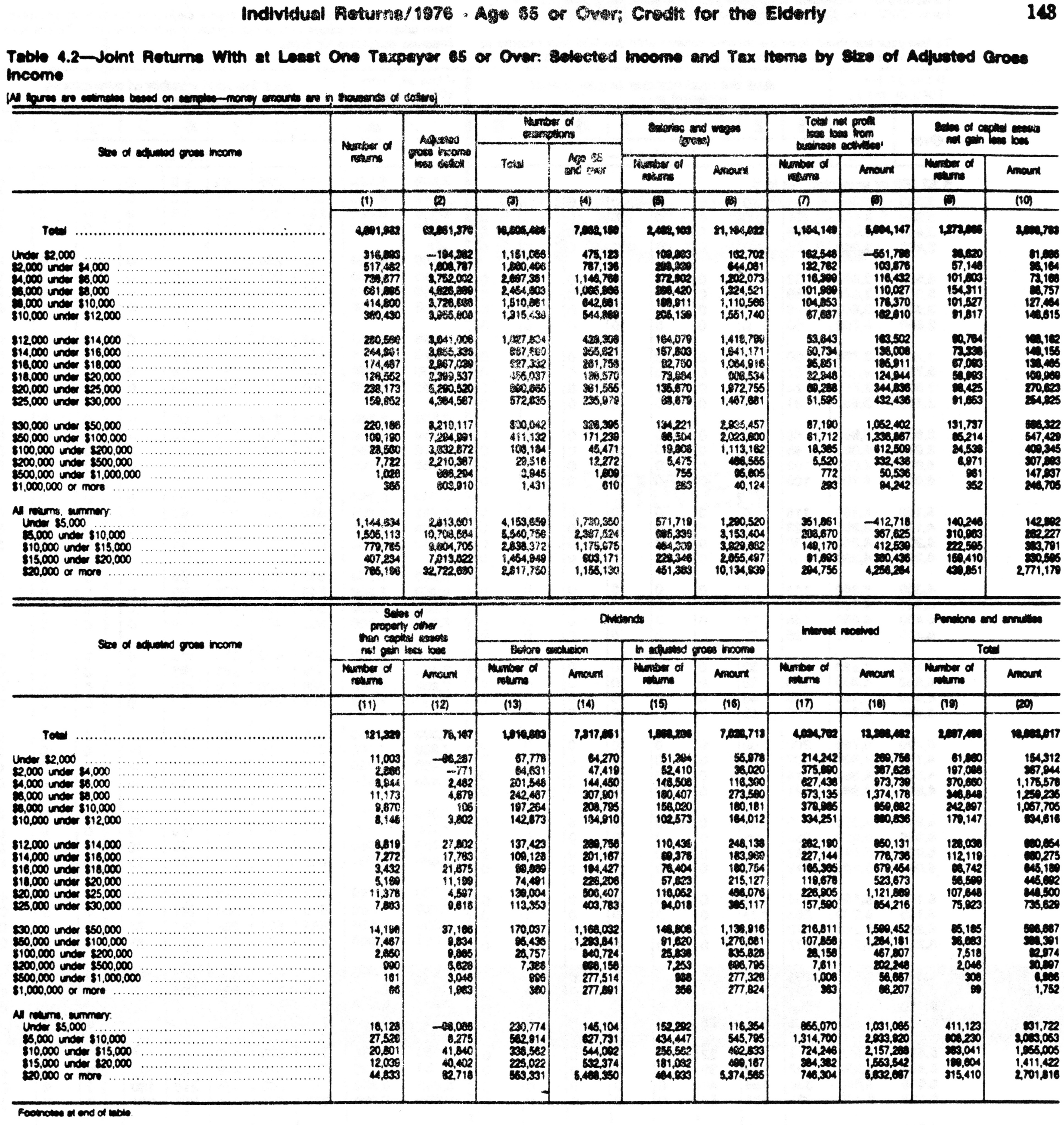
(1) The Treasury Enforcement Communications System (TECS) is a Treasury-wide data and communications system that links Treasury law enforcement field terminals with their National Offices and with each other. It also provides direct contact with the FBI's National Crime Information Center (NCIC), the National Law Enforcement Telecommunications System (NLETS), the Telephone Number Analysis System (TELAN) and Customs Service terminals located at land-border crossings, key ports of entry, and international airports.

(2) The Service has twelve TECS terminals in operation. One located at the National Office, Criminal Investigation Division; one in the National Office, Internal Security Division, Investigations Branch; and one in each of the offices of the Chief, Criminal Investigation Branch at service center locations.

ATTACHMENT 12 - KEY WORD IN CONTEXT PRINTOUT

```
CHAPTER   PAGE   LINE

INTERNAL  FORM 4797     627  WITH RESPECT TO SUCH PROPERTY SHALL BE TREATED AS|ALLOCABLE TO SUCH PORTION TO THE EXTENT OF THE ARC
INTERNAL  FORM 4797     630  WITH RESPECT TO SUCH PROPERTY SHALL BE TREATED AS|ALLOCABLE TO SUCH UNDIVIDED INTEREST TO THE EXTENT
INTERNAL  FORM 4874     803  IARIES ON THE BASIS OF THE ESTATE OR TRUST INCOME|ALLOCABLE TO EACH.  IN THE CASE OF A SMALL BUSINES
INTERNAL  FORM 4625    1273  S INCOME FROM SUCH PROPERTIES LESS THE DEDUCTIONS|ALLOCABLE TO THEM, EXCEPT FOR EXCESS INTANGIBLE DP

INTERNAL  FORM 4684    1398  10, AND IF APPLICABLE IN PART III, COLUMN H (I). |ALLOCATE THE$ 100 LIMITATION FOR EACH CASUALTY OR

INTERNAL  FORM 3468     251  IABILITY LIMITATION (SECTION 46 (A) (3)) SHALL BE|ALLOCATED TO THE PATRONS OF THE COOPERATIVE.  6. 6
INTERNAL  FORM 4562    1207  ION 167 (K): (A) LIST REHABILITATION EXPENDITURES|ALLOCATED TO THAT UNIT (SEE I.T. REGS. 1.167 (K) -

INTERNAL  FORM 3468     260   MUST ATTACH TO THIS FORM A STATEMENT SHOWING THE|ALLOCATION OF THE INVESTMENT AMONG ITS BENEFICIARI
INTERNAL  FORM 4136     657  RETURN, FORM 1065.  THE STATEMENT SHOULD SHOW THE|ALLOCATION AMONG THE PARTNERS OF THE GALLONS USED
INTERNAL  FORM 4562    1213  AL AT WHICH EACH VACANT UNIT IS OFFERED.  (10) IF|ALLOCATION IS REQUIRED UNDER I.T. REGS. 1.167 (K)

INTERNAL  SCHEDULE S    137  Y AS DEFINED IN SECTION 46 (A) (4) REDUCED BY THE|ALLOWABLE CREDIT FOR REGULAR INVESTMENT CREDIT PRO
INTERNAL  SCHEDULE S    139  AND IS THEN ENTERED ON FORM 3468 TO DETERMINE THE|ALLOWABLE CREDIT FOR THE YEAR (SEE INSTRUCTION FOR
INTERNAL  SCHEDULE S    192  PECTIVE RETURNS.  THE REFUNDABLE ENERGY CREDIT IS|ALLOWABLE ONLY FOR TAX YEARS ENDING AFTER SEPTEMBE
INTERNAL  FORM 3468     294  EDIT YOU CANNOT USE BECAUSE IT EXCEEDS THE AMOUNT|ALLOWABLE (INCLUDING ANY UNUSED CREDIT CREATED BY
INTERNAL  FORM 3468     310  REASE YOUR QUALIFIED INVESTMENT FOR A YEAR BY THE|ALLOWABLE QUALIFIED INVESTMENT IN PROGRESS EXPENDI
INTERNAL  FORM 3468     396   THE ADDITIONAL 1 PERCENT CREDIT ON LINE 5 (A) IS|ALLOWABLE TO THE EXTENT THE CORPORATION MAKES THE
INTERNAL  FORM 4797     399  NGE) IN COLUMN D.  ENTER DEPRECIATION ALLOWED (OR|ALLOWABLE) IN COLUMN E AND THE COST OR OTHER BASIS
INTERNAL  FORM 4797     463  R: (A) DEPRECIATION OR AMORTIZATION IS ALLOWED OR|ALLOWABLE AFTER CERTAIN DATES (SEE SECTIONS 1245 A
INTERNAL  FORM 4797     493  Y INCOME TO THE EXTENT OF DEPRECIATION ALLOWED OR|ALLOWABLE AFTER 1961.  FOR FARM PROPERTY SUBJECT T
INTERNAL  FORM 4797     511  CTS.  LINE 18 (A).  ENTER DEPRECIATION ALLOWED OR|ALLOWABLE AFTER DECEMBER 31, 1961 (JUNE 30, 1963,
INTERNAL  FORM 4797     513  AY BE REQUIRED TO INCLUDE DEPRECIATION ALLOWED OR|ALLOWABLE ON ANOTHER ASSET IF THE ADJUSTED BASIS O
INTERNAL  FORM 4797     617  ME.  INCLUDE ON LINE 15 ANY DEPLETION ALLOWED (OR|ALLOWABLE) IN DETERMINING THE ADJUSTED BASIS OF TH
INTERNAL  FORM 4874     822     IF THE CREDIT FOR THE TAX YEAR IS MORE THAN THE|ALLOWABLE AMOUNT, THE UNUSED PORTION MAY BE CARRIE
INTERNAL  FORM 3903     863  XPENSES FROM YOUR FORMER TO YOUR NEW RESIDENCE IS|ALLOWABLE FOR ONLY ONE TRIP.  HOWEVER, IT IS NOT N
INTERNAL  FORM 3903     895  1040 OR RELATED SCHEDULES.  M. NO DOUBLE BENEFITS|ALLOWABLE.  YOU CANNOT REDUCE THE AMOUNT OF GAIN O
INTERNAL  FORM 4562    1142  IS COST OR OTHER BASIS BY DEPRECIATION ALLOWED OR|ALLOWABLE IN PRIOR YEARS AND ITS SALVAGE VALUE.  D
INTERNAL  FORM 4562    1179   PARTNERSHIP SHARE OF ASSETS COST IN FIGURING THE|ALLOWABLE DEDUCTION.  ADDITIONAL FIRST-YEAR DEPREC
INTERNAL  FORM 4625    1235  HEDULE A, FORM 1040, LINE 29).  (3) ANY DEDUCTION|ALLOWABLE UNDER SECTION 691 (C).  STEP 2.  MULTIPL
INTERNAL  FORM 4625    1239  E APPROPRIATE LINE (5) THE EXCESS OF DEPRECIATION|ALLOWABLE OVER THE DEPRECIATION THAT WOULD HAVE BE
INTERNAL  FORM 4625    1239  OWABLE OVER THE DEPRECIATION THAT WOULD HAVE BEEN|ALLOWABLE IF YOU HAD USED THE STRAIGHT LINE METHOD
INTERNAL  FORM 4625    1244  ECT TO A LEASE.  ENTER THE EXCESS OF DEPRECIATION|ALLOWABLE OVER THE DEPRECIATION THAT WOULD HAVE BE
INTERNAL  FORM 4625    1245  OWABLE OVER THE DEPRECIATION THAT WOULD HAVE BEEN|ALLOWABLE IF YOU HAD USED THE STRAIGHT-LINE METHOD
INTERNAL  FORM 4625    1248  D (G), ENTER THE AMOUNT BY WHICH THE AMORTIZATION|ALLOWABLE IS MORE THAN THE DEPRECIATION DEDUCTION
INTERNAL  FORM 4625    1248  IS MORE THAN THE DEPRECIATION DEDUCTION OTHERWISE|ALLOWABLE.  IF YOU USE THE CLASS LIFE ASSET DEPREC
INTERNAL  FORM 4625    1250  NES 1 (B) AND 1 (C).  ANY VARIANCE IN USEFUL LIFE|ALLOWABLE UNDER SECTION 167 (M) (1) IS ALSO CONSID
INTERNAL  FORM 4625    1254    IN FIGURING THE DEPRECIATION DEDUCTION OTHERWISE|ALLOWABLE.  (M) RESERVES FOR LOSSES ON BAD DEBTS O
INTERNAL  FORM 4625    1256  TO THE RESERVE FOR BAD DEBTS THAT WOULD HAVE BEEN|ALLOWABLE IF YOU HAD MAINTAINED THE BAD DEBT RESER
INTERNAL  FORM 4625    1260  ION.  ENTER ANY EXCESS OF DEDUCTION FOR DEPLETION|ALLOWABLE UNDER SECTION 611 OVER THE PROPERTY'S AD
INTERNAL  FORM 4625    1269  NGIBLE DRILLING COSTS ARE THE AMOUNT BY WHICH THE|ALLOWABLE INTANGIBLE DRILLING AND DEVELOPMENT COST
INTERNAL  FORM 4625    1270  TIVE WELL) EXCEED THE AMOUNT THAT WOULD HAVE BEEN|ALLOWABLE IF SUCH COSTS HAD BEEN CAPITALIZED AND T
INTERNAL  FORM 4684    1350  RIOR TAXABLE YEAR, PROVIDED THE LOSS IS OTHERWISE|ALLOWABLE AS A DEDUCTION FOR THE YEAR IN WHICH IT
INTERNAL  FORM 4684    1373  T PLUS IMPROVEMENTS LESS DEPRECIATION (ALLOWED OR|ALLOWABLE), AMORTIZATION, DEPLETION, ETC. (SEE SEC

INTERNAL  FORM 3903     892  TO COMMENCE WORK.  INCLUDE ALL REIMBURSEMENTS AND|ALLOWANCES FOR MOVING EXPENSES IN INCOME.  IN GENE
INTERNAL  FORM 3903     893  ENERAL, FORM W-2 INCLUDES SUCH REIMBURSEMENTS AND|ALLOWANCES.  HOWEVER, CHECK WITH YOUR EMPLOYER IF
INTERNAL  FORM 3903     894  BT.  USE LINE 12 FOR REPORTING REIMBURSEMENTS AND|ALLOWANCES IF THEY ARE NOT INCLUDED ELSEWHERE ON F
INTERNAL  FORM 4562    1153   NOT RESULT AT THE END OF ANY YEAR IN ACCUMULATED|ALLOWANCES THAT ARE MORE THAN THE AMOUNT THAT COUL

INTERNAL  SCHEDULE S    153  RN.  ENERGY PROPERTY MEANS PROPERTY WHICH IS: (1)|ALTERNATIVE ENERGY PROPERTY, (2) SOLAR OR WIND ENE
INTERNAL  SCHEDULE S    161   DETAILS.  THE FOLLOWING SPECIAL RULES APPLY: (1)|ALTERNATIVE ENERGY PROPERTY, SOLAR OR WIND ENERGY
```

ATTACHMENT 13 - TABLE LAYOUT SAMPLE ONE



Individual Returns/1976 · Age 65 or Over; Credit for the Elderly 148

Table 4.2—Joint Returns With at Least One Taxpayer 65 or Over: Selected Income and Tax Items by Size of Adjusted Gross Income

[All figures are estimates based on samples—money amounts are in thousands of dollars]

Size of adjusted gross income	Number of returns	Adjusted gross income less deficit	Number of exemptions		Salaries and wages (gross)		Total net profit less loss from business activities¹		Sales of capital assets net gain less loss	
			Total	Age 65 and over	Number of returns	Amount	Number of returns	Amount	Number of returns	Amount
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
Total	**4,891,982**	**62,861,376**	**16,805,486**	**7,882,189**	**2,492,103**	**21,194,032**	**1,164,149**	**5,884,147**	**1,273,866**	**3,898,783**
Under $2,000	316,893	−194,282	1,151,066	475,123	109,883	162,702	162,548	−551,798	36,620	81,886
$2,000 under $4,000	517,482	1,808,787	1,880,496	787,136	288,939	644,061	132,762	103,876	57,148	96,164
$4,000 under $6,000	739,877	3,752,002	2,697,361	1,146,769	372,802	1,202,073	116,399	116,432	101,803	73,168
$6,000 under $8,000	661,895	4,626,889	2,454,803	1,065,868	288,420	1,324,521	101,989	110,027	154,311	86,757
$8,000 under $10,000	414,800	3,726,698	1,510,861	642,661	189,911	1,110,566	104,853	176,370	101,527	127,464
$10,000 under $12,000	360,430	3,955,608	1,315,439	544,869	205,139	1,551,740	67,687	162,610	91,817	148,615
$12,000 under $14,000	280,596	3,641,006	1,027,804	429,308	164,079	1,418,799	53,643	163,502	80,764	169,182
$14,000 under $16,000	244,861	3,655,336	867,560	355,821	157,803	1,641,171	50,734	136,008	73,338	148,155
$16,000 under $18,000	174,487	2,967,039	627,332	261,758	92,750	1,064,916	35,851	186,911	67,093	138,465
$18,000 under $20,000	126,552	2,399,537	456,037	186,570	73,884	808,534	32,848	124,944	58,893	109,969
$20,000 under $25,000	238,173	5,290,520	860,666	361,565	135,670	1,872,755	69,288	344,836	98,425	270,823
$25,000 under $30,000	159,952	4,364,587	572,835	235,979	88,679	1,467,681	51,595	432,436	91,653	254,825
$30,000 under $50,000	220,186	8,210,117	800,042	328,395	134,221	2,934,457	87,190	1,052,402	131,737	896,322
$50,000 under $100,000	109,190	7,284,991	411,132	171,239	86,504	2,023,800	61,712	1,336,867	85,214	547,429
$100,000 under $200,000	28,580	3,832,872	108,184	45,471	19,808	1,113,162	18,365	612,509	24,538	409,345
$200,000 under $500,000	7,722	2,210,387	29,516	12,272	5,475	466,555	5,520	332,438	6,971	307,893
$500,000 under $1,000,000	1,028	686,294	3,945	1,809	755	95,605	772	50,536	961	147,937
$1,000,000 or more	365	803,910	1,431	610	263	40,124	293	94,242	352	246,705
All returns, summary:										
Under $5,000	1,144,834	2,813,601	4,153,659	1,730,360	571,719	1,290,520	351,861	−412,718	140,246	142,892
$5,000 under $10,000	1,505,113	10,708,664	5,540,758	2,387,524	895,339	3,153,404	208,670	367,625	310,963	282,227
$10,000 under $15,000	779,765	9,804,705	2,838,372	1,175,675	484,309	3,829,862	149,170	412,539	222,595	383,791
$15,000 under $20,000	407,234	7,013,622	1,454,949	603,171	229,346	2,855,497	81,693	380,436	169,410	330,595
$20,000 or more	765,196	32,722,680	2,817,750	1,155,130	461,383	10,134,939	294,756	4,256,264	439,851	2,771,179

Size of adjusted gross income	Sales of property other than capital assets net gain less loss		Dividends				Interest received		Pensions and annuities	
			Before exclusion		In adjusted gross income				Total	
	Number of returns	Amount	Number of returns	Amount	Number of returns	Amount	Number of returns	Amount	Number of returns	Amount
	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)
Total	**121,329**	**76,167**	**1,916,883**	**7,317,861**	**1,898,206**	**7,028,713**	**4,034,762**	**13,398,482**	**2,097,488**	**10,883,017**
Under $2,000	11,003	−96,287	67,778	64,270	51,394	56,978	214,242	269,758	61,860	154,312
$2,000 under $4,000	2,868	−771	64,631	47,419	52,410	36,020	375,890	387,628	197,088	367,944
$4,000 under $6,000	8,944	2,482	201,548	144,450	148,508	116,300	627,438	973,739	370,660	1,175,578
$6,000 under $8,000	11,173	4,679	242,487	307,901	180,407	273,580	573,135	1,374,178	348,848	1,259,236
$8,000 under $10,000	9,870	105	197,264	208,795	156,020	180,181	379,985	859,682	242,897	1,057,705
$10,000 under $12,000	8,146	3,802	142,873	194,910	102,573	164,012	334,251	880,836	179,147	934,616
$12,000 under $14,000	8,819	27,802	137,423	289,758	110,436	248,138	282,190	850,131	128,038	880,654
$14,000 under $16,000	7,272	17,763	109,128	201,167	89,376	183,969	227,144	776,736	112,119	880,275
$16,000 under $18,000	3,432	21,875	98,869	194,427	76,404	180,754	165,365	679,454	86,742	845,189
$18,000 under $20,000	5,189	11,189	74,491	226,206	57,823	215,127	119,678	523,673	56,599	445,692
$20,000 under $25,000	11,378	4,597	139,004	506,407	116,052	486,076	226,905	1,121,869	107,648	848,500
$25,000 under $30,000	7,883	9,818	113,353	403,783	94,018	395,117	157,590	854,216	75,923	736,629
$30,000 under $50,000	14,199	37,166	170,037	1,168,032	148,808	1,138,916	216,811	1,599,452	85,165	596,687
$50,000 under $100,000	7,467	9,834	96,436	1,293,841	91,620	1,270,681	107,856	1,284,181	36,683	398,391
$100,000 under $200,000	2,850	9,865	26,757	840,724	25,838	835,828	28,158	467,807	7,518	82,974
$200,000 under $500,000	990	6,628	7,388	898,158	7,253	896,795	7,611	202,248	2,046	30,897
$500,000 under $1,000,000	161	3,046	996	277,514	988	277,328	1,008	56,667	308	6,906
$1,000,000 or more	66	1,963	360	277,891	356	277,824	363	66,207	99	1,752
All returns, summary:										
Under $5,000	16,128	−98,066	230,774	145,104	152,292	116,354	866,070	1,031,085	411,123	831,722
$5,000 under $10,000	27,520	8,275	562,914	627,731	434,447	545,795	1,314,700	2,933,920	808,230	3,083,053
$10,000 under $15,000	20,801	41,840	338,562	544,092	256,562	492,833	724,246	2,157,288	393,041	1,956,005
$15,000 under $20,000	12,039	40,402	225,022	532,374	181,092	499,187	384,382	1,563,542	199,604	1,411,422
$20,000 or more	44,833	82,718	563,331	5,468,350	484,933	5,374,565	746,304	6,832,667	315,410	2,701,816

Footnotes at end of table.

ATTACHMENT 14 - TABLE LAYOUT SAMPLE TWO

1979 Tax Table D—HEAD OF HOUSEHOLD (Filing Status Box 4)

For unmarried (including certain married persons who live apart and abandoned spouses) or legally separated persons who qualify as heads of household with income of $20,000 or less on line 34, Form 1040, or line 11, Form 1040A, who claim 8 or fewer exemptions.

To find your tax: Read down the income column until you find your income as shown on line 34, Form 1040, or line 11, Form 1040A. Read across to the column headed by the total number of exemptions claimed. The amount shown where the two lines meet is your tax.

The $2,300 zero bracket amount and your deduction for exemptions have been taken into account in figuring the tax shown in this table. **Do not take a separate deduction for them.**

If line 34, Form 1040, or line 11, Form 1040A, is—		And the total number of exemptions claimed is—							
Over	But not over	1	2	3	4	5	6	7	8
		Your tax is—							
If $3,300 or less your tax is 0									
3,300	3,350	4	0	0	0	0	0	0	0
3,350	3,400	11	0	0	0	0	0	0	0
3,400	3,450	18	0	0	0	0	0	0	0
3,450	3,500	25	0	0	0	0	0	0	0
3,500	3,550	32	0	0	0	0	0	0	0
3,550	3,600	39	0	0	0	0	0	0	0
3,600	3,650	46	0	0	0	0	0	0	0
3,650	3,700	53	0	0	0	0	0	0	0
3,700	3,750	60	0	0	0	0	0	0	0
3,750	3,800	67	0	0	0	0	0	0	0
3,800	3,850	74	0	0	0	0	0	0	0
3,850	3,900	81	0	0	0	0	0	0	0
3,900	3,950	88	0	0	0	0	0	0	0
3,950	4,000	95	0	0	0	0	0	0	0
4,000	4,050	102	0	0	0	0	0	0	0
4,050	4,100	109	0	0	0	0	0	0	0
4,100	4,150	116	0	0	0	0	0	0	0
4,150	4,200	123	0	0	0	0	0	0	0
4,200	4,250	130	0	0	0	0	0	0	0
4,250	4,300	137	0	0	0	0	0	0	0
4,300	4,350	144	4	0	0	0	0	0	0
4,350	4,400	151	11	0	0	0	0	0	0
4,400	4,450	158	18	0	0	0	0	0	0
4,450	4,500	165	25	0	0	0	0	0	0
4,500	4,550	172	32	0	0	0	0	0	0
4,550	4,600	179	39	0	0	0	0	0	0
4,600	4,650	186	46	0	0	0	0	0	0
4,650	4,700	193	53	0	0	0	0	0	0
4,700	4,750	200	60	0	0	0	0	0	0
4,750	4,800	207	67	0	0	0	0	0	0
4,800	4,850	214	74	0	0	0	0	0	0
4,850	4,900	221	81	0	0	0	0	0	0
4,900	4,950	228	88	0	0	0	0	0	0
4,950	5,000	235	95	0	0	0	0	0	0
5,000	5,050	242	102	0	0	0	0	0	0
5,050	5,100	249	109	0	0	0	0	0	0
5,100	5,150	256	116	0	0	0	0	0	0
5,150	5,200	263	123	0	0	0	0	0	0
5,200	5,250	270	130	0	0	0	0	0	0
5,250	5,300	277	137	0	0	0	0	0	0
5,300	5,350	284	144	4	0	0	0	0	0
5,350	5,400	291	151	11	0	0	0	0	0
5,400	5,450	298	158	18	0	0	0	0	0
5,450	5,500	306	165	25	0	0	0	0	0
5,500	5,550	314	172	32	0	0	0	0	0
5,550	5,600	322	179	39	0	0	0	0	0
5,600	5,650	330	186	46	0	0	0	0	0
5,650	5,700	338	193	53	0	0	0	0	0
5,700	5,750	346	200	60	0	0	0	0	0
5,750	5,800	354	207	67	0	0	0	0	0
5,800	5,850	362	214	74	0	0	0	0	0
5,850	5,900	370	221	81	0	0	0	0	0
5,900	5,950	378	228	88	0	0	0	0	0
5,950	6,000	386	235	95	0	0	0	0	0
6,000	6,050	394	242	102	0	0	0	0	0
6,050	6,100	402	249	109	0	0	0	0	0
6,100	6,150	410	256	116	0	0	0	0	0
6,150	6,200	418	263	123	0	0	0	0	0
6,200	6,250	426	270	130	0	0	0	0	0
6,250	6,300	434	277	137	0	0	0	0	0
6,300	6,350	442	284	144	4	0	0	0	0
6,350	6,400	450	291	151	11	0	0	0	0
6,400	6,450	458	298	158	18	0	0	0	0
6,450	6,500	466	306	165	25	0	0	0	0
6,500	6,550	474	314	172	32	0	0	0	0
6,550	6,600	482	322	179	39	0	0	0	0
6,600	6,650	490	330	186	46	0	0	0	0
6,650	6,700	498	338	193	53	0	0	0	0
6,700	6,750	506	346	200	60	0	0	0	0
6,750	6,800	514	354	207	67	0	0	0	0
6,800	6,850	522	362	214	74	0	0	0	0
6,850	6,900	530	370	221	81	0	0	0	0
6,900	6,950	538	378	228	88	0	0	0	0
6,950	7,000	546	386	235	95	0	0	0	0
7,000	7,050	554	394	242	102	0	0	0	0
7,050	7,100	562	402	249	109	0	0	0	0
7,100	7,150	570	410	256	116	0	0	0	0
7,150	7,200	578	418	263	123	0	0	0	0
7,200	7,250	586	426	270	130	0	0	0	0
7,250	7,300	594	434	277	137	0	0	0	0
7,300	7,350	602	442	284	144	4	0	0	0
7,350	7,400	610	450	291	151	11	0	0	0
7,400	7,450	618	458	298	158	18	0	0	0
7,450	7,500	626	466	306	165	25	0	0	0
7,500	7,550	635	474	314	172	32	0	0	0
7,550	7,600	644	482	322	179	39	0	0	0
7,600	7,650	653	490	330	186	46	0	0	0
7,650	7,700	662	498	338	193	53	0	0	0
7,700	7,750	671	506	346	200	60	0	0	0
7,750	7,800	680	514	354	207	67	0	0	0
7,800	7,850	689	522	362	214	74	0	0	0
7,850	7,900	698	530	370	221	81	0	0	0
7,900	7,950	707	538	378	228	88	0	0	0
7,950	8,000	716	546	386	235	95	0	0	0
8,000	8,050	725	554	394	242	102	0	0	0
8,050	8,100	734	562	402	249	109	0	0	0
8,100	8,150	743	570	410	256	116	0	0	0
8,150	8,200	752	578	418	263	123	0	0	0
8,200	8,250	761	586	426	270	130	0	0	0
8,250	8,300	770	594	434	277	137	0	0	0
8,300	8,350	779	602	442	284	144	4	0	0
8,350	8,400	788	610	450	291	151	11	0	0
8,400	8,450	797	618	458	298	158	18	0	0
8,450	8,500	806	626	466	306	165	25	0	0

Continued next column

Continued on next page

GCA

GENCODE™

PRIMER

GCA

Graphic Communications Association
1730 North Lynn Street / Suite 604 • Arlington, Virginia 22209 • Phone (703) 841-8160

An Affiliate of Printing Industries of America, Inc.

[BLANK PAGE]

GenCode™ Primer
A Concept of Generic Text Markup

Prepared and modified over the years
by members of the
GCA GenCode™ Committee
Revised through 1981

GCA grants to CBEMA, ANSI, and ISO permission royalty free, worldwide, irrevocably to quote from this document with attribution © (i.e., annotating the quoted material with a notice of its protection and owner thereof), and to reproduce, publish, and distribute for all purposes of the standardization process, and to anyone anywhere permission to use any issue standard.

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

GenCode™ Primer

Table of Contents

I. Introduction 171
  A. Purpose of this Primer 171
  B. Generic Coding 171
  C. GenCode™ 171
II. How GenCode™ Works 172
  A. Structure 172
    1. Documents 172
    2. Components 173
      a. Zones 173
      b. Segments 173
      c. Elements 173
    3. Interrelationships of Elements 175
      a. Sequential Elements 175
      b. Asequential Elements 175
      c. Bounded Elements 175
      d. Nonnested Elements 175
      e. Nested Elements 175
  B. Syntax 175
III. Implementation Considerations 177
  A. GenCode™ Approaches 177
  B. Processing Requirements 177

List of Figures

1. General Document Hierarchy 172
2. Sample Zone and Segment Hierarchy 173
3. Sample Recursive Hierarchy 174
4. Sample Element Hierarchy 174

List of Tables

1. Zones 173
2. Segments 173
3. Elements 173

List of Examples

1. Asequential Element 175
2. Bounded Element 175
3. Nested Elements 176

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

I. INTRODUCTION

A. Purpose of this Primer

The purpose of this publication is to introduce the reader to GenCode™, GCA's concept for generic coding of text files. This primer provides an account of the benefits of GenCode™ and a description of its uses and requirements. Examples are included throughout to clarify the meaning. Other publications will be released to detail GenCode's™ implementation and to define options referenced in this primer.

It is hoped that readers will be stimulated to promote or conduct tests of the adaptability of GenCode™ to their own particular requirements. GCA is anxious to learn the results of wide-ranging implementation of GenCode™ and requests that readers forward their observations and comments to GCA to aid in refinements and extensions of the GenCode™ system.

B. Generic Coding

Generic coding is a data management technique used to identify text components of a text data file in such a way that the text itself is not linked to any particular processing device or application. Because the file is coded for its intrinsic content instead of its relationship to a product, any one file can be used for many different byproducts and by many different processing systems. To put it concisely: ***THE DATA IS INDEPENDENT OF ITS APPLICATION.*** By the use of generic coding, the data in a text file can be processed in whole or in part by a multitude of different application programs. The necessary translations and transformations to compensate for differing computer languages and recording media are not considered major obstacles to the implementation of processing.

Generic coding, in various forms, is already in use today. Some companies use it to reduce the cost of document preparation, others to enhance retrievability. Another benefit which accrues from the principle of "separating the data from the application" is portability between different systems. These techniques are in use primarily at large-volume shops, but as the costs of computerization continue to plummet, and as generic coding and its benefits become more widely known, the trend to create generically coded text files will accelerate.

Of course, for generic coding to be functional, the file must be coded in a systematic, logical way. Each data component must be defined: For example, in a data processing environment, typical data components of a personnel data record might include **EMPLOYEE NAME, SOCIAL SECURITY NUMBER** and **GROSS PAY.** In a text file, data components might include **AUTHOR NAME, TITLE, HEADING,** and **PARAGRAPH.** The interrelationships of components to each other must also be defined in a generic file. Marking up and structuring the data in this way is the basis of generic coding.

C. GenCode™

For generic coding to work on a national basis within the entire data processing industry, rules must be made for all to follow. With no central set of standards by which everyone can communicate, the main purpose of generic coding is defeated. The GCA has assumed the responsibility to develop a standard system of generic coding to handle the increased volume of text data being captured. The GCA's implementation of generic coding is called GenCode™.

GenCode™ is a *mark-up language* which is used *to identify the data* in a file. It is not a *processing language* which is used *to format data.* The GenCode™ markup language also identifies the intrinsic, logical structure of a file.

GenCode™ is a practical approach to text markup suitable for today's needs and open-ended enough to permit future expansion for tomorrow's needs. By the use of GenCode™, authors and editors are supported in their preparation of a manuscript in machine-readable form without regard to output product specifications so that text data files can be transferred between systems and still retain their structural integrity. Output images are created by associating a generic code with the typographical or display requirements for each generic component of that file.

GenCode™ *does not* entail a specific definition of output requirements or a rigid file structure. Output processing quality need not be reduced because there is no need to restrict the capabilities of individual systems. Vendors and users of GenCode™ can maintain the "uniqueness of approach" of their own systems, and no additional support facilities which exceed the user needs are required.

Anyone producing information products with computerized text processing equipment directly benefits from the use of GenCode™; suppliers and consumers of these products benefit indirectly. Vendors benefit from an enhanced ability to serve a broader array of customers, increased market opportunity, and greater competitiveness. Authors benefit from greater control over their product and enhanced acceptability of their manuscript from a wider selection of publishers. Publishers benefit from multiple data use and greater vendor independence. Information consumers benefit from increased access to information in various forms.

Few of these benefits will be realized if every user and vendor develops his own generic coding technique; it is for that reason that GCA is proposing this particular technique known as GenCode™.

In case of conflict between any statement in any article and the standard, the standard governs.

II. HOW GenCode™ WORKS

Before GenCode™ can be understood and applied, the *structure* of a data file and the *syntax* of the markup language must be defined as used in GenCode™. GenCode's™ main code is a generic identifier (**GI**), which defines the structure of the data and which must be applied by certain simple syntactical rules.

A. Structure

In generic coding, the structure of a file must be delineated. GenCode™ adopts a basic, simplified structure definition (see Figure 1) which each user can refine to suit his or her own purposes.

1. Documents

The largest unit GenCode™ defines is a Document. A Document may be any self-contained text string, from a simple interoffice memo to a complex book. It may be made up of sub-Documents or may in turn be made a part of another larger Document. This idea of Documents contained within Documents is called *recursion*. To illustrate, the *Chicago Style Manual* consists of several parts each of which has its own front matter and body matter. This is a recursive hierarchy. Data can be coded in such a way that

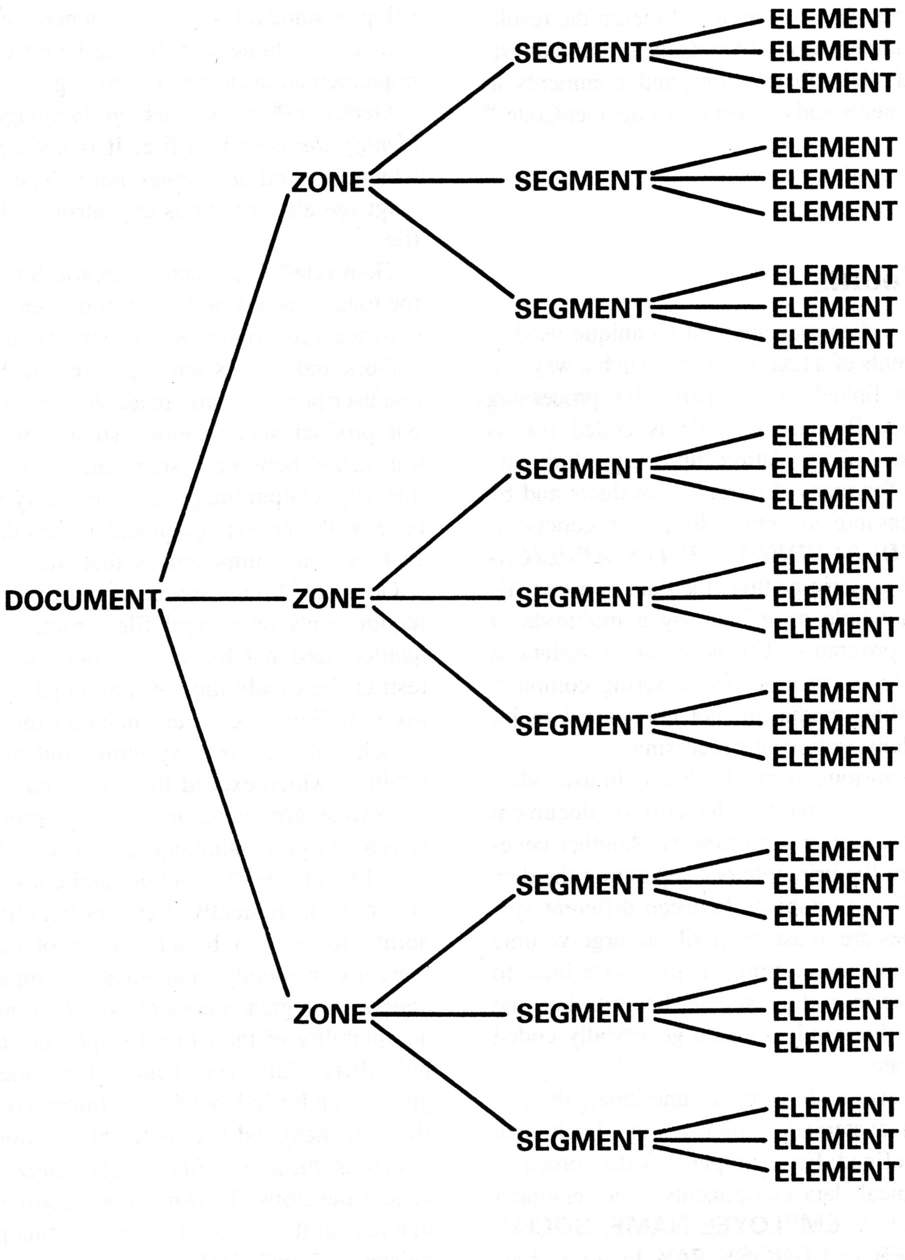


FIGURE 1. General Document Hierarchy

In case of conflict between any statement in any article and the standard, the standard governs.

the coding indicates whether the data is always a Document, may be used as a Document, or can never be a Document.

The **GI** for the Document itself is the first item of generic coding to be encountered. All other **GI**'s used may be unique to and created for that Document within the conventions of the GenCode™ structure. It is this flexibility in establishing **GI**'s and their placement that gives GenCode™ its virtually universal applicability.

2. Components

Within any given Document, GenCode™ identifies textual data components and hierarchies by the use of individual or combined precedence codes associated with the following three GenCode™ components: Zones, Segments, and Elements.

a. Zones

A document is comprised of one or more major units called Zones. These Zones are the next step down in the hierarchy (see Figure 2) from the Document itself. If the Document is a book, the Zones are front matter (title page, copyright page, etc.), body matter (chapter, part, section, etc.), and rear matter (appendix, index, etc.) All books contain body matter, but need not include front matter or rear matter. Body matter is always necessary for all Documents no matter what their form.

The division of a Document into Zones establishes primary keys for invoking output formatting tables used in a translation process to a specific application. These keys may also be used in retrieving or extracting textual elements contained in each of the Zones. Established **GI**'s for Zones are listed in Table 1 although it is not necessary to use them.

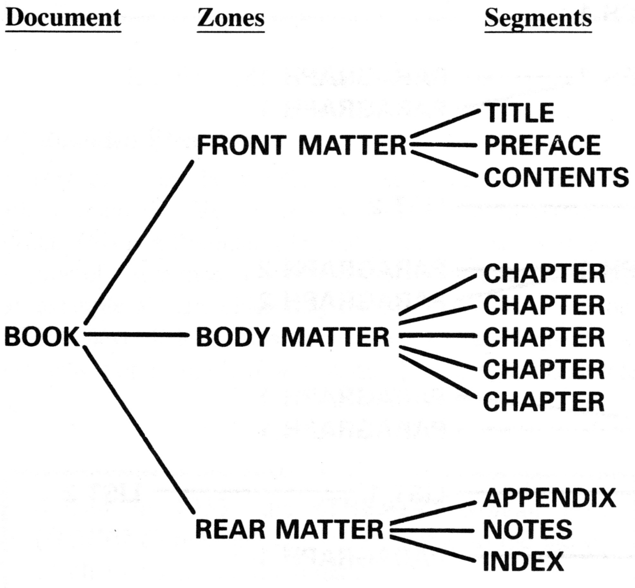


FIGURE 2. Sample Zone and Segment Hierarchy

Zone	*GI*
Front Matter	**FM**
Body Matter	**BM**
Rear Matter	**RM**

TABLE 1. Zones

b. Segments

The information content and structure of a document file is further defined by combining each Zone component with Segment components. Segments specify Zone subsets such as preface, section, chapter, index, etc. Segments are the next step down in the hierarchy (see Figure 2).

Segments should be delineated and defined in the manner most practical to fit a user's requirements for his or her particular data. A publisher of textbooks might, for example, specify a Segment **CHAP** for chapter segments, whereas the publisher of an encyclopedia or dictionary might specify a segment **ALPH** for alphabetical break segments. Table 2 is a listing of sample **GI**'s for Segments.

Segment	*GI*
Title	**TITL**
Copyright	**COPY**
Part	**PART**
Chapter	**CHAP**
Tabular Material	**TABL**
Table Box	**TBOX**
Index	**INDX**

TABLE 2. Segments

c. Elements

An Element is the GenCode™ component needed to further identify text data and define more complex hierarchies. Textual matter in this category includes internal heads, paragraphs, lists, etc. Figure 3 is an example of a possible hierarchical structure including Elements. Table 3 is a sample list of **GI**'s for Elements.

Element	*GI*
Paragraph	**P1, P2,** etc.
List	**L1, L2,** etc.
Tabular Entry	**T1, T2,** etc.

TABLE 3. Elements

In case of conflict between any statement in any article and the standard, the standard governs.

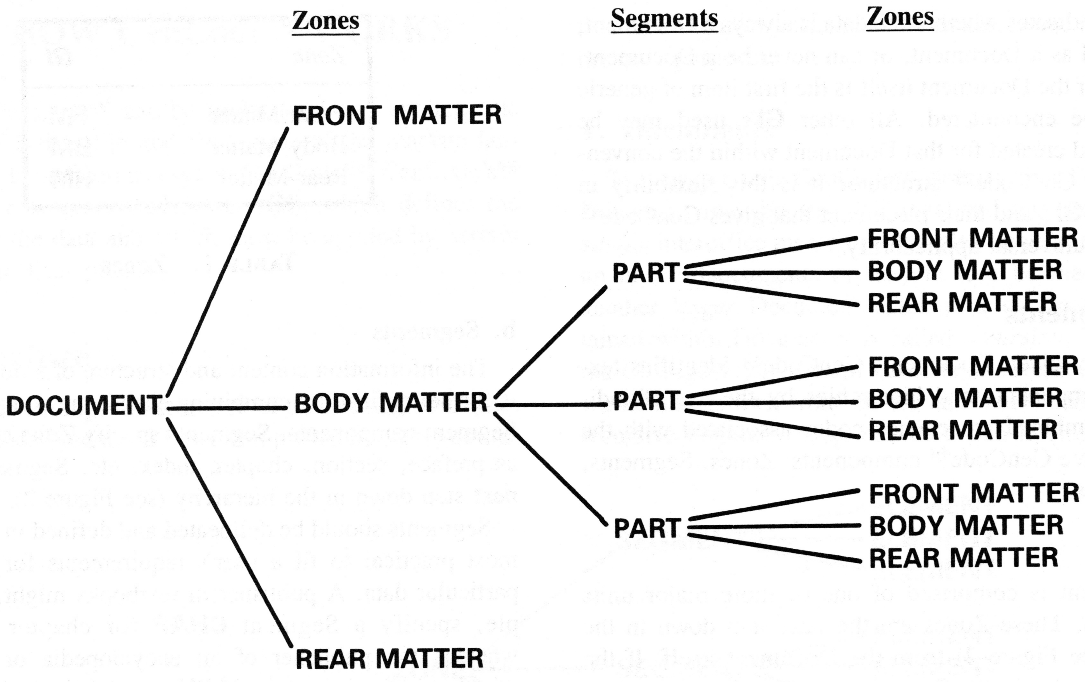

Zones
Segments
Zones
FRONT MATTER
DOCUMENT
BODY MATTER
REAR MATTER
PART
PART
PART
FRONT MATTER
BODY MATTER
REAR MATTER
FRONT MATTER
BODY MATTER
REAR MATTER
FRONT MATTER
BODY MATTER
REAR MATTER

FIGURE 3. Sample Recursive Hierarchy

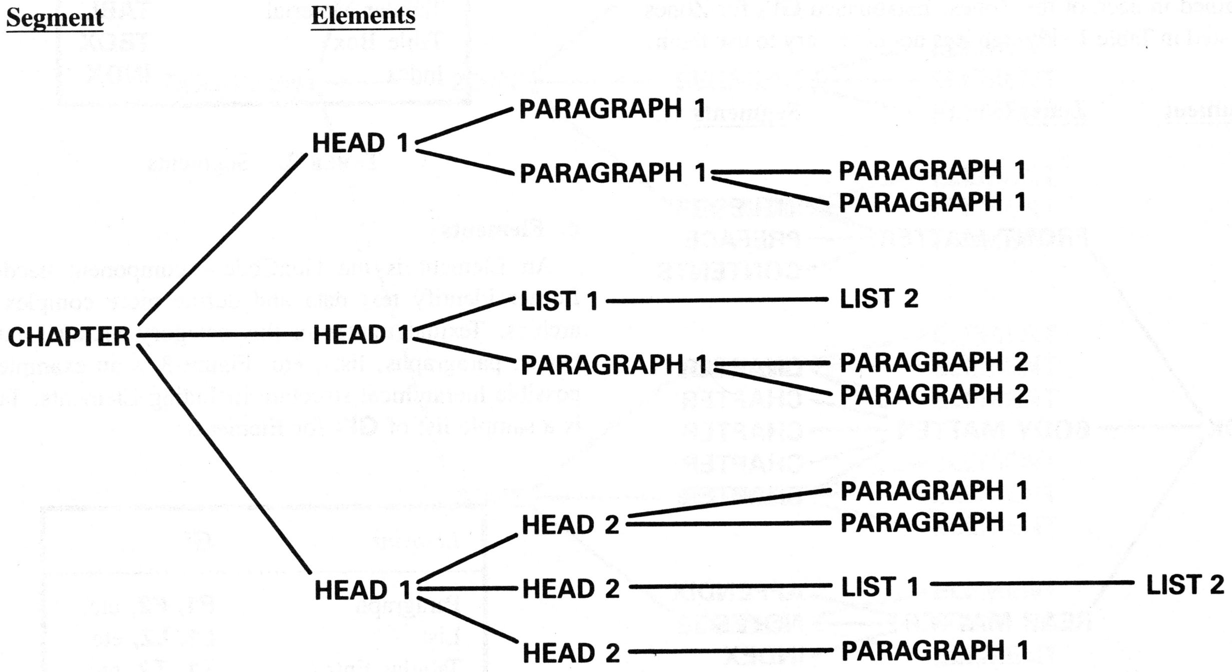

Segment
Elements
CHAPTER
HEAD 1
PARAGRAPH 1
PARAGRAPH 1
PARAGRAPH 1
PARAGRAPH 1
HEAD 1
LIST 1
LIST 2
PARAGRAPH 1
PARAGRAPH 2
PARAGRAPH 2
HEAD 1
HEAD 2
PARAGRAPH 1
PARAGRAPH 1
HEAD 2
LIST 1
LIST 2
HEAD 2
PARAGRAPH 1

FIGURE 4. Sample Element Hierarchy

In case of conflict between any statement in any article and the standard, the standard governs.

3. Interrelationships of Elements

Elements can be related in intricate ways. Among GenCode's™ strengths is the ability to describe any type of data organization. The following is a discussion of various element interrelationships.

a. Sequential Elements

A Sequential Element is an Element of data which must appear in the sequential order implied by the information which precedes and follows it in the file. For example, paragraphs are sequential. They must remain in the order they are input into the file for the information content to remain intact.

b. Asequential Elements

Asequential Elements identify text which need not appear in the sequential order implied by the information which precedes and follows it in the data file. For example, a footnote may appear directly after its text reference, but on the final output product that footnote data could, and probably does, appear at some other point. Example 1 is a sample of an Asequential Element.

> The Chief Justice, whose office is established in the Constitution, **⟨FNRS⟩** Article I, Section 3, Clause 6 **⟨/FNRS⟩** is head of the Supreme Court.
>
> *or*
>
> The Chief Justice, whose office is established in the Constitution, **⟨FNR RID = EX1⟩** is head of the Supreme Court. **⟨FN ID = EX1⟩** Article I, Section 3, Clause 6. **⟨/FN⟩**

EXAMPLE 1. Asequential Element

c. Bounded Elements

Text data may be wholly enclosed within another text Element on the file but intended to appear in sequential order. This is a Bounded Element. A court case cited in the middle of a sentence or a reference call-out for a footnote is an example of data which may need coding to set it off but which, nevertheless, must remain in its exact sequence in the data. For a sample of a Bounded Element, see Example 2.

> Signed in 1953, the **⟨TREATY⟩** Korean Pact **⟨/TREATY⟩** pledges the U.S. to come to the aid of South Korea in certain situations.

EXAMPLE 2. Bounded Element

d. Nonnested Elements

An Element which is not subordinate to another Element is a nonnested Element.

e. Nested Elements

Certain Elements are, by their nature, subordinate to other Elements. These subordinate Elements are said to be nested within other Elements. For example, a file may have a series of heads, some of which are subheads of another head. Similarly, the paragraphs of an extract may be subordinate to the text. Conditions like these need special coding to indicate their subordination. In Example 3, the intricacies of nesting are shown.

B. Syntax

As in any language, if communication is to take place, certain syntactical conventions must be followed. GenCode™ has a set of simple rules of syntax which make it possible for all data processing systems which follow those rules to read and interpret a GenCoded™ file easily.

The syntactical rules of GenCode™ are as follows:

(1) The code must begin with a "less than" (⟨) opening delimiter and end with a "greater than" (⟩) closing delimiter.

(2) The **GI** must follow the opening delimiter and may consist of from one to eight characters arbitrarily selected by the user. It is suggested that frequently used **GI**'s be short for ease of use while less frequently used **GI**'s be longer for ease of remembering their identification purpose.

(3) One or more optional attribute codes can follow the **GI** enclosed within the same opening and closing delimiters. Attributes do not have a defined sequence within the code and are of two types: a *self-defining value* and a *labeled value*. A self-defining value attribute consists of from one to eight characters and has an inherent meaning. A labeled value consists of a label from one to eight characters to give a meaning to a value followed by an equal sign and the value assigned to that attribute. Attributes must be separated by a space from the **GI** and from each other if there is more than one attribute.

(4) Any **GI** can have an ending identifier signalling the end of a component. The ending code has the same opening and closing delimiters and **GI** as the opening code with a slash between the opening delimiter and the **GI**. Attributes do not have to be included in a closing code, although they may be. An asequential, bounded, or nested Element must have a closing code

In case of conflict between any statement in any article and the standard, the standard governs.

Manuscript Format

The following is an outline of terms:

I. Capitalism is an economic doctrine based upon private ownership of the means by which goods and services are produced.

II. A democracy exists where the power to govern is held by the people themselves, where government is conducted by, and with the consent of the people.

Modern democracies are indirect—that is, representative democracies—rather than direct (or pure) in form. Democratic thought insists upon belief in

A. The fundamental worth of every individual
B. The equality of all men
C. Majority rule and minority rights
D. Individual freedom

III. Socialism is a philosophy of economic collectivism, advocating public ownership of the major instruments of production, distribution, and exchange.

Evolutionary socialists are to be distinguished from revolutionary socialists; most of the latter are communists.

Studying those concepts will help the student gain a better idea of world governments.

GenCode™ Application

```
⟨P1⟩ The following . . .
⟨LIST⟩ [first level list]
⟨L1⟩ Capitalism is . . .
⟨L1⟩ A democracy exists . . .
⟨P1⟩ Modern democracies . . .
⟨LIST⟩ [second level list]
⟨LI⟩ The fundamental . . .
⟨L1⟩ The equality . . .
⟨L1⟩ Majority rule . . .
⟨L1⟩ Individual freedom . . .
⟨/LIST⟩ [end second level list]
⟨L1⟩ Socialism . . .
⟨P1⟩ Evolutionary . . .
⟨/LIST⟩ [End first level list]
⟨P1⟩ Studying . . .
```

Example 3. Nested Elements

In case of conflict between any statement in any article and the standard, the standard governs.

indicating the end of the Element. If an asequential or bounded Element encounters a code for a sequential or unbounded Element, a closing code is not strictly necessary. If a nested Element encounters a code for a nonnested Element or a higher level Element, the lower level Element is ended.

III. IMPLEMENTATION CONSIDERATIONS

A. GenCode™ Approaches

Basically, GenCode™ may be created in two environments—author and editorial.

There are two types of author environments. The author environment may be one in which the author assigns GenCode™ to raw text independent of any consideration for planned production procedures or ultimate processing formats, or it may be one in which the author is given direction on code assignment by associate production personnel. For the purpose of explanation, these conditions are termed *author-independent* and *author-directed,* respectively.

In the author-independent—or simplified—environment, coding involves assignment of appropriate Element codes devoid of Zones and Segments. For example, **⟨H1⟩** is used for a code as opposed to the more complex **⟨BM⟩⟨CHAP⟩⟨H1⟩** which is assigned in an author-directed approach. The simplified approach allows an author to identify textual Elements while ignoring output processing considerations, but it also necessitates incorporation of the latter information at some later point in the production process.

The author-directed approach is obviously desirable, since it minimizes editorial efforts. It also reflects a cooperative preplanning effort between the author and production personnel in terms of document formats, hierarchical structures, and production processes.

The editorial environment is one in which GenCode™ is applied to raw text either during or prior to keyboard entry, or one in which code application is an automated operation associated with the extraction of textual data from data processing files or neutral data bases.

Both the author and editorial environments are applicable in the creation of entry data for GenCode™ data bases associated with data base management systems.

B. Processing Requirements

The ideal system is one which can recognize GenCode™ markup codes directly and assign format functions to the file at the point of the code. The next best alternative is a translation program which replaces a GenCode™ markup code with one or more corresponding application codes and transforms the generic file to a specific application file format.

In order for a file to be converted in this way, a table of relationships between the GenCode™ markup code and the application code must be built in machine-readable form. The data required for this table can be supplied in machine-readable form or just in written form to the recipient of the file. The recipient then uses whatever methods are available to him or her to create the table in the format required by his or her conversion program.

In the case of a simple conversion process on a system with limited capabilities, it may be necessary for the user to perform manually the task of code conversion through his or her own text editing system. In this case, the table of relationships need not be established in machine-readable form.

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

GENCODE™

EXPOSITORY MATERIAL

Graphic Communications Association
1730 North Lynn Street / Suite 604 • Arlington, Virginia 22209 • Phone (703) 841-8160

An Affiliate of Printing Industries of America, Inc.

[BLANK PAGE]

GenCode™ Expository Material

Table of Contents

1. **"An Introductory Rationale for the Standard Generalized Markup Language,"** 1983 June 15 183

C. "Kit" von Sück
Marketing Specialist
Publishing & Broadcast Industries
Digital Equipment Corporation

2. **"A Short Introduction to Office Applications of Markup Languages—Using SGML,"** 1983 April 15 187

James D. Mason
Text Management Specialist
Oak Ridge National Laboratory

3. **"Generic Codes,"** 1978 193

A collective effort under the leadership of:

Brian L. Murray, Director
Text Editing & Photocomposition Branch
John A. Sullivan, Head
Composition Software Section
Canadian Government Printing Office

4. **"Full-Text Data Base Publishing,"** 1983 May 12 201

Donald H. Rollert
Manager, Graphic Systems Development Division
U.S. Government Printing Office

5. **"Integrating Text and Graphics,"** 1982 213

Aaron H. Bigman
Senior Financial Publishing Consultant
Standard & Poor's Corporation

6. **"Rules & Regulations Via GenCode™,"** 1983 May 12 221

Joseph P. Johnson
Technology Assessment Officer
Federal Deposit Insurance Corporation

7. **"GenCode™ Preprocessor Requirements,"** 1983 June 15 .. 227

Jordan J. Glogau
Executive Vice President
Medianet, Inc.

In case of conflict between any statement in any article and the standard, the standard governs.

GenCode™ Expository Material (*Continued*)

8. **''Document Processing at Boeing Computer Services,''** 1983 June 15 243

Sharon Adler
Manager, Production Services
Boeing Computer Services

9. **''Comparison of Generic, Mnemonic, and Native-Mode Coding,''** 1983 May 3 247

Joseph V. Gangemi
Vice President, Systems and Programming
Lehigh/Rocappi, Inc.

An Introductory Rationale for the Standard Generalized Markup Language (SGML)

C. "Kit" von Sück

Marketing Specialist
Publishing & Broadcast Industries
Digital Equipment Corporation

Member		
ANSI	Task Group	X3J6.1
EG CLPT	Task Group	SGML

Material prepared for this publication
1983 June 15

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the GCA copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Graphic Communications Association. To copy otherwise, or to republish, requires a fee and specific permission. © 1983 GCA

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

For you, the reader of this standard, to accept the ideas enclosed and the concepts that are presented, you must first agree that there is a *compelling* need for generic markup. To illustrate that need, imagine for a moment that you are the manager of a large publications department. Your job is challenging and rewarding, but lately you have had a large increase in your group's work load.

You have a large main frame running your customized publishing system. The system allows you to output formatted text to either a high-speed line printer, a laser printer, or a phototypesetter. As long as your "documents" are input with the proper codes, your system can make up pages and output the text. You have had the system for several years, and you can make it sit up and do undocumented tricks. It is a good system for your work.

Two problems have started to occur in your department, though. The first problem is that multiple authors of the same document are using different "keying conventions." (One author uses a one-inch indent for paragraphs; another uses no indent.) The list of differences has your editors and proof readers ready for a mutiny. They are beginning to spend more time correcting format than worrying about content. Their work output has been dropping, and several of your key people are out looking for new jobs.

The second problem involves taking information from a subcontractor on the project who is using a different publishing system. The subcontractor refuses or cannot afford to implement your system just for this one joint project. You therefore agree to take tapes that contain "text only" and add formatting commands after the text is received.

A. What Can You Do?

You have a real problem trying to coordinate the activities of the technical writers/authors, both your own and the subcontractor's. You have asked for opinions, talked to various people, and come away with one general conclusion: "If I could get everybody to do it the same way, all my problems would disappear."

Your idea is sound. But now you wonder "How in the world am I ever going to get twenty writers to adhere to the rules and learn a new language for this project?" You wrote a good User's Guide, only ten pages long with excellent examples. The subcontractor's writers, however, refuse to learn a new system. And furthermore, your technical writers feel it is not their job to worry about the output, just their technical accuracy. They tell you "Format it yourself if you don't like the way we do it now!"

Suppose there was a way to separate the content of the project from the format. Support there was a system compatible with your present one and your subcontractor's, even though they are made by different companies. Suppose your editors didn't have to worry about format either, they just worried about accuracy and continuity.

Although this may seem like a lot to ask for, the Standard Generalized Markup Language (SGML) will do these tasks for you. *SGML separates form from content* by keeping each TEXT ELEMENT in a document separate (e.g., paragraphs, lists and individual list items, figures, voice, etc.).

A key principle in the use of SGML is to separate formatting from content. This is accomplished by stating in the text stream where each text element begins and ends. This can be either explicit, such as "**⟨Begin Paragraph⟩**" and "**⟨End Paragraph⟩**", or implicit. (In SGML the angle brackets, or less-than and greater-than symbols, are used to delineate markup from text.)

An example of implicit text element might be "two carriage returns equals the end of one paragraph and the beginning of another." The paragraph has been separated into a logical section, but without any apparent markup. And no markup is the best kind!

B. The Solution

While it is easy to say that a system will solve all of your problems. It is quite another problem to define that system in simple terms. While the standard itself may be hard to interpret, *the concepts are easy to understand.*

Every text element has a logical beginning and a logical end. If each element can be separated, it can be processed to produce the desired results. Further, the person who is entering the text should not have to be concerned with making the final version presentable. Let the author worry about content, the graphic designer will handle "the looks".

This differs with most present-day systems where the person entering the text formats the output at the same time. Word processors need rulers, more complex systems like SCRIPT, RUNOFF, NROFF, and most publication systems imbed system-unique commands into the text stream.

It is very hard for us to break old habits. One might argue that they would like to see a bulleted list like this:

Here is item one,

Here is item two, and

So on.

rather than the generic equivalent, which could look like this:

```
⟨BLIST⟩
⟨ITEM⟩    Here is item one,
⟨ITEM⟩    Here is item two, and
⟨ITEM⟩    So on.
⟨/BLIST⟩
```

However, **if you were to design the logic to process the first example, it would be far more complex than would be necessary for the generic example.** It should be noted, too, that a listing in example one is not a logical element, just a collection of text preceded with a lower case "O". In contrast, a listing in the second example is clearly marked as a bulleted list, with clear beginning and end points.

In case of conflict between any statement in any article and the standard, the standard governs.

While present text processing systems are good at what they do, there are some limitations. Chief among these drawbacks is interchange with other, dissimilar systems. Generic markup eliminates system-dependent commands, makes passing of documents for remote processing a reality. Text can be sent from one site to another with confidence that the remote site can read and understand both the structure model and the contents of a document.

As anyone who has tried to process data received from an outside source knows, the first major task to undertake is data conversion from one processing language into the language of your machine/system. **Generic markup will eliminate this time-consuming process.** In fact, one of the first items in a generically coded document is the **STRUCTURE** command that describes the hierarchy and the names of the **GENERIC IDENTIFIERS** used in the document that follows.

One of the other principles of generic markup is that you, as a writer/author should be able to name your own elements, that is to create your own start-tag and end-tag for each element in a document. For instance, you might choose to use a **⟨P⟩** to start a new paragraph, where someone else might choose to use a **⟨PARA⟩.** Neither approach is wrong, that is why it is considered generic.

This frees the author from the worry of compliance with publishers' restrictions for submission of text, or the problems of multiple authors mentioned above. Al long as you are **consistent within a document,** any name may be used for a tag. Again, the vehicle for knowing what the element names used in the document are is the **STRUCTURE** declaration, or **STRUC.**

The output of this approach will be consistent and correct because the author did not try to format the text; he/she just entered the initial key strokes. Every "**⟨P⟩**" paragraph will be indented the same amount when passed to the output processor.

In case of conflict between any statement in any article and the standard, the standard governs.

A Short Introduction to Office Applications of Markup—Using SGML

Dr. James D. Mason

Text Management Specialist
Oak Ridge National Laboratory

Member

ANSI		X3J6
ANSI	Task Group	X3J6.1
EG CLPT	Task Group	SGML
ANSI		X3V1
ANSI	Secretary	X3V1.3

Special Notes

1. This paper was presented first at the meeting of ISO/TC97/SC18, Text Preparation and Interchange, Working Group 3, Text Structures, in Paris, April 1983. It has also been used by ANSI X3V1, Office Systems, Task Group 3, as their document 83-13.

2. Oak Ridge National Laboratory, Oak Ridge, Tennessee, is operated by Union Carbide Corporation's Nuclear Division under contract No. W-7405-eng-26 for the U.S. Department of Energy.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the GCA copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Graphic Communications Association. To copy otherwise, or to republish, requires a fee and specific permission. © 1983 GCA

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

The approaching publication of a Draft International Standard for Computer Languages for Processing of Text may make designers of office systems wonder what relevance it has for their applications. It has been rumored that the designers of this standard, being oriented heavily towards publishing systems, have produced something that is alien to the office and can be used only by programmers, not secretaries. Perhaps, then, it is time to clear up some of the misunderstandings about the proposed standard and to examine how it might be applied in office systems.

In the first place, the world of publishing is not necessarily so different from that of the office. The basic machine of the office, the nominal model for most word-processing systems, is the typewriter. Yet the very name of that device indicates that it attempts to adapt a basic technique of publishing—setting type—to a function essential to both offices and publishing houses—writing. Furthermore, many of the new concepts currently fashionable among office-systems people originated in the worlds of publishing and large-scale computing: telecommunications, local-area networks, electronic mail. Among the most automated offices in the world are those of newspapers, where large-scale electronic publishing brought computer terminals to replace the typewriters of reporters and editors, bridging the gap between offices and typesetting.

In the second place, the world of publishing, whether it be of books, reports, periodicals, or newspapers, is not isolated from the world of the office, its personnel, and its machinery. Copy for publication originates with writers (or their typists) and their typewriters, word processors, and personal computers. Today it would be very hard for a publisher of any size to ignore the idea so commonly called "capturing original keystrokes." Even though the planners of the proposed document-markup standard are, for the most part, involved in large-scale publishing, they are also involved in the work of the office, for they represent users (and also manufacturers) of office systems in their daily work.

The proposed markup standard is a set of tools for describing the structures of documents—of *all* types. To this end, the tools have been made independent of any particular documents or processing systems. The markup, along with the text of a document, may be typed in by human operators. It may be generated by function keys on some systems. In some cases it may even be inferred from the input by an intelligent processor. Although nominally designed for direct use by electronic text-management systems, there is nothing in the standard to preclude text (including all markup) intended for use on such a system from being keyed on simple typewriters and only later entered into a system by such means as optical character-recognition devices. While the standard is intended to be character-set independent (it provides mechanisms for bootstrapping character sets other than a default), it does not require added control functions or code extension for its implementation.

Nor is the markup designed solely for preparing conventional copy for viewing or reproduction: the same techniques could be used to describe text structures for data-base management, indexing and retrieval, linguistic analysis, and many other applications. Furthermore, the standard does not distinguish between systems in which the user views the unprocessed, coded text and those in which the user views only the results of the coding.

How might such a standard work in an office environment? One could hardly find a more basic office task than memo creation to take as an example. But to show how the standard allows extension of functionality, the example will go one step further to an electronic mail system, since such systems are often patterned after paper memos.

On a centralized, multipurpose office system, creation of a message for electronic mail might be initiated simply by typing a typical opening line:

```
To: Willard Rodgers .
```

The system might respond with

```
From:
```

and expect the user to supply a name. The dialogue might continue, asking for a subject and the intended recipients of copies and blind copies. It would probably verify names of addressees from a central data base as they are entered. Other information, such as holding time for the message, might also be requested. When finished, the dialogue might look like this (with the user's entries in italic type):

To: *Willard Rodgers*
From: *Jim Mason*
Copies: *Donna Griffith, AnnaJo Shelton*
Blind copies: *Jim Mason*
Subject: *Demonstration Message* .

At this point, the system's normal text-entry program could take over.

What the user saw on the text-entry screen might be only the text of the message, but a sophisticated system should make it possible to edit the results of the initial dialogue, in case the user found an error in the subject or decided to change the list of intended recipi-

In case of conflict between any statement in any article and the standard, the standard governs.

ents. In that case, the user should be presented with the results of the system's processing. Since, for the purpose of the example, the markup system is taken in its basic form, the display might begin with the dialogue input surrounded by codes inserted by the system:

```
<to>Willard Rodgers<\to>
<from>Jim Mason<\from>
<copies>Donna Griffith, AnnaJo Shelton<\copies>
<blindcopies>Jim Mason<\blindcopies>
<subject>Demonstration Message<\subject> .
```

(The system could also have responded with free-form text

```
<to>Willard Rodgers<\to><from>Jim
Mason<\from><copies>Donna
Griffith, AnnaJo Shelton<\copies><blindcopies>
Jim Mason<\blindcopies>
<subject>Demonstration Message<\subject>
```

since the processing application can be triggered by the markup delimiters alone. But that is not a particularly friendly display, so it will just be mentioned in passing.) In each case the elements of text have been labeled with *generic identifiers*, and each element has both *starting markup* and *ending markup.*

This kind of coding is likely to be unfamiliar to users of conventional word-processing systems, on which codes are generated by function keys and are generally hidden from the operator. But it is familiar to many users of typesetting systems, though the individual codes may be unusual. Perhaps an office system would not wish to display something that looks so much like "typesetting coding," but before showing how it can be eliminated, the example will follow it through a few more steps.

Actually, there is more coding here than is necessary. The system can be implemented so that a simplified *ending markup* indicates the end of the current element:

```
<to>Willard Rodgers<\>
<from>Jim Mason<\>
<copies>Donna Griffith, AnnaJo Shelton<\>
<blindcopies>Jim Mason<\>
<subject>Demonstration Message<\> .
```

So far, the coded example has every element delimited at both beginning and end. But frequently that is not necessary. One of the important features of the markup system is *markup minimization.* If the applications processor knows something about the structure of the documents it is to handle—that certain elements always appear in a unique sequence or that a particular one of them cannot appear within another—much of the coding can be eliminated. In the present example, it might be logical to assume that the entry for the recipients of copies, if it appears, must follow the entry for the primary recipients and cannot be part of it. With this kind of document logic, the information stored, processed, and displayed might be reduced to a slightly less intimidating form:

```
<to>Willard Rodgers
<from>Jim Mason
<copies>Donna Griffith, AnnaJo Shelton
<blindcopies>Jim Mason
<subject>Demonstration Message .
```

The beginning of each new structural element also terminates the previous element, so that *ending markup* can be eliminated.

Even this kind of display, with something that looks like coding or at least probably is not what will go in a message as displayed to a recipient (or in a printed memo, since it could be generated from the same coding), is not always necessary. The system could easily follow the example of many word processors by displaying something other than what it stores. In short, the system would interpret the markup and format the text so that the display might look more like the original dialogue:

```
To: Willard Rodgers
From: Jim Mason
Copies: Donna Griffith, AnnaJo Shelton
Blind copies: Jim Mason
Subject: Demonstration Message .
```

A moderately intelligent processor could scan this message header section, discover whether changes have been made in it, and replace the markup tags in their proper locations before actual transmission or printing of the message. The only major requirement for the processor is knowledge of message structure. This example shows an important feature of the system: markup can be used for multiple applications. Here markup is used both to format the display and the message delivered to the recipient and to find the address information for the delivery system.

This is a relatively simple example, but it shows the basic method by which markup is entered into text and

In case of conflict between any statement in any article and the standard, the standard governs.

the kind of application that might be developed for it. There are more aspects of markup, such as how predefined objects, like special characters, graphics, or even other complete documents, can be brought into a document under preparation (*entity references*). The system allows definition of character sets, selection of delimiters (although the defaults are graphic characters, there is no prohibition against making them control characters or code extensions), and inclusion of processing commands from the host system. Markup entries can carry much more information than the examples have shown, since it is possible to qualify markup with numerous *attributes* beyond simple marking of the structure of a document.

Among the most important aspects of the system is the ability to define document structures. In the simplest implementations, the structures of basic document types will probably be hard coded: there will be a set of formatting procedures, and the markup processor will simply recognize the markup tags and call the appropriate procedures. For advanced systems, however, the standard proposes that the markup processor be able to read a document-structure declaration (*struc*) that indicates which elements can appear under what circumstances in a document. With this capability, both document structure and the relationship of markup to structure should be dynamically redefinable. Although the *attributes* attached to the *generic identifiers* can be used for such functions as modifying formats or indicating *rank* for elements such as headings, there must ultimately be procedures supplied by a processing program. A system capable of dynamic modification of document processing would need, of course, a rather large repertoire of procedures available for the markup processor to call.

For the mail or memo header examined above, the document structure could be represented with a *struc* declaration such as that shown in Exhibit 1.

```
<!STRUC --
ELEMENT      MIN?      CONTENT   --
message                header,body
header       O    O    to, from, copies?, blindcopies?,
                       subject
to           -    O    namelist
from         -    O    namelist
copies       -    O    namelist
blindcopies  -    O    namelist
namelist     O    O    name, (",",  " "?*, name)?*
name         O    O    nametext
subject      O    O    copytext
>
```

Exhibit 1. *Struc* declaration for a message system.

The declaration might continue with definitions of text suitable for names, subjects, and other elements of the message body. What this structure command indicates is that a message consists of a header and a body. The message header consists of a "to" block followed by a "from" block, optional "copies" and "blindcopies" blocks, and a "subject" block. Each of the names defined here becomes a *generic identifier* for the document type "message." The first four of these generic blocks in the header all have the same structure, that of a "namelist" which consists of a name, optionally followed by a repeatable continuation sequence of a comma, any number of spaces, and another name. Definitions of "nametext" and "copytext" could be made as simple or complex as the system allows or the application requires. The "MIN" column indicates what sort of *markup minimization* is possible for each type of item. Since this is a known sequence, the end of each item can be signaled by the beginning of the next, and *ending markup* is not needed. A hyphen in this column shows that minimization is not allowed; "O" indicates simple omission of the tag (there are several other types of minimization possible besides omission). The identifier "header" need not be used in actual markup, since it is known to be the first item in a message and it is terminated by the beginning of the body. Accordingly, neither *starting markup* nor *ending markup* is necessary. Similarly, "namelist" and "name" are used only as part of a structural definition and need not appear in actual markup. The markup for "name" and "subject" can be eliminated if they are properly defined. It is this *minimization* that makes the simplified coding shown in the examples possible. Structure declarations for the body text of the message would follow the same pattern.

As this *struc* table shows, the markup processor would need relatively little power for a message or memo system. The information needed to perform the *markup minimization* is essentially all the structural knowledge required. For message transfer the processor would use the "namelist" items to extract delivery information from the mail system's data base. For presentation, the processor might know only that each entry should appear on a new line (though it probably should also be prepared to handle an entry that needs more space, like a long distribution list).

The processing requirements of a system to implement the markup standard are not excessive. The mechanisms for maintaining generic structures, such as automatically numbered headings and enumerations, are already available on commercial word processors. All that is necessary for such systems to implement the proposed standard for interchange is to insert appropriate markup into the output text stream or recognize it in the input stream and convert it to their

In case of conflict between any statement in any article and the standard, the standard governs.

internal codes. In each case a relatively simple translator is all that is needed.

Other applications are already available. A number of commercial electronic mail systems are implemented along the lines of the example given above. Similar programs for memo creation have been installed on a number of systems. An example of a fairly complete generic markup system, lacking only a new input parser to recognize inline markup, is *pwbmm,* the *Programmer's Workbench Memorandum Macros* from Bell Laboratories, available on many UNIX systems. This system, which includes extensive facilities for headings, lists, footnotes, cover pages and tables of contents for technical reports, with most of the formatting parameters dynamically redefinable during processing, occupies only about 1400 lines of high-level code, and most of that is for error checking and internal bookkeeping.

The great advantages of such a system are mostly in its extensibility. It can handle a variety of document types, with new structures being available at relatively little expense. No new control codes are needed. Multiple nested documents are processed without great difficulty. It can be added to existing document-preparation systems. Document transfer is simplified because there is no need for system-dependent codes or specialized record types. It can easily handle input from a variety of sources. Although originally implemented on systems where coding is typed directly into the system by an operator, the coding could easily be produced by the processing system and never made visible to the user.

An additional advantage for the standards community is that the research for standards preparation is almost complete. What is needed now is not so much a new system of document coding as applications and definitions for implementation of the markup language on new types of systems.

In case of conflict between any statement in any article and the standard, the standard governs.

Generic Codes

Brian L. Murray

Manager
Text Editing & Photocomposition Branch
Canadian Government Printing Office

John A. Sullivan

Head
Composition Software Section
Canadian Government Printing Office

Led a collective effort which resulted
in the generation and publication of
this material.

This material, originally identified as ''HUGO,'' was prepared circa 1978.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the GCA copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Graphic Communications Association. To copy otherwise, or to republish, requires a fee and specific permission. © 1983 GCA

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

I. CONCEPT OF GENERIC CODING

The concept of GENERIC CODING originated from the need of having a tool to represent identification of text data in such a way that standardization for a common utilization would be available.

Any document is a means of transferring information, whether it be a series of ideas or several thoughts, from the author to the reader. To clearly represent, separate and emphasize these ideas, several methods are used: The division into parts, sections or chapters, the use of paragraphs and sentences.

In an author's work, whether it is a report, a list, a catalogue or even a novel, the ideas flow, complement and introduce one another; there is a natural train of thought. In other words, an author's work contains an INHERENT HIERARCHY and STRUCTURE, and any document representing this author's work must also represent this hierarchy and structure.

GENERIC CODING is a technique for coding textual data file by representing a document structure and by identifying all the levels of information within its hierarchy. Other characteristics, uses and advantages of GENERIC CODES are described below.

(1) It supports author's preparation of manuscript without regard to ultimate typographic specifications, which aids in reduction of publisher production cost.

(2) It provides a common means of textual data exchange between users.

(3) It provides a common identification of textual data elements for information retrieval, composition or other application.

(4) It is a tool to enter text in a structured and uniform method.

(5) Overcome major obstacles to data exchange between system users.

(6) It permits more flexibility in text data manipulation.

(7) A document needs no further supportive information.

(8) It is a self-contained unit of information from which not only the structure of the document can be learned, but also all the pertinent information about each piece of text within the document.

(9) By following the natural hierarchy and structure of the document, it is easy to learn and use.

II. PARTS OF PUBLICATION

A publication is divided into three distinctive parts. These are:

(1) Preliminary matter

(2) Text or main body

(3) Back pages, sometimes called the reference matter.

Each part contains different sections and a breakdown of said parts is as follows:

A. Preliminary Matter

The preliminary matter is usually arranged in the following sequence although there are no set rules:

1. Book Half Title.

Usually the book half title is the first printed page in a book. Only the name of the book is printed on this page. Half titles are sometimes also used before each part of the text if broken into parts and depending on the layout, before each chapter.

2. Title Page

The title page generally includes the following:

(1) Title: It covers the contents adequately but briefly;

(2) Author: May include author's name, or department, etc.;

(3) Place of publication;

(4) Date of publication;

(5) Publisher;

(6) Serial number or Volume number;

(7) Price, if any.

On the back of the title page there could appear such information as:

(1) Number and date of a revision or printing;

(2) Name of the printer and copyright notice.

3. Dedication

When there is a dedication page, it appears on a separate page immediately after the title page.

4. Foreword and Preface

Although these terms are to some extent synonymous, a distinction is made between them.

The foreword is a general introduction discussing the field covered and stressing the value of the work and the standing of the author. In governmental publication, it is sometimes replaced by a formal letter of transmittal from a minister or senior official.

The preface is reserved for the author's introduction to his own publication. It may consist of a brief statement to the purpose of the publication, a general summary of the method of presenting the material and an acknowledgement of assistance from other individuals, departments or organizations.

This also is the first page that is numbered, not necessarily the first counted page. It is numbered with a roman numeral.

5. Table of Contents

This page is a brief outline of the content of the book and it should omit any reference to the pages up to and including the table of contents.

6. List of Illustrations

The style or form of this page usually follows the table of contents.

7. Introduction

In government publications, material that would otherwise be included in a preface (but do not) appear as the first few paragraphs of the text.

B. Text or Main Body

This part of a book contains the main purpose of the work.

C. Back Pages

These pages may contain any of the following:

(1) appendix

(2) references

(3) abbreviations

(4) glossary

(5) bibliography

(6) index

(7) colophon.

A set of codes with a glossary of possible application and meaning is found hereafter.

III. LIST OF GENERIC CODES

Generic codes can be either at the beginning of a paragraph or on a line by themselves.

They start with an * and end by a carriage return if they are a line by themselves. If they are at the beginning of a paragraph, they can be anywhere on a line and they must start with an * and end by a space.

They are used to determine the following:

(1) Parts of publication;

(2) Parts of text - Chapters and Headings;

(3) Parts of text - Paragraphs;

(4) Other features commonly used.

A. Code Definitions or Descriptions Parts of Publication

***cover**	Use to indicate the matter used on the outside or front of a book (Possible subjects treated on cover are book title, book number, date of book, author and the subject matter).
***halftitle**	Use for the half title page contains only the name of the book, can also be used before each part of text and sometimes before each chapter.
***title**	That part of a book which includes some of the following: TITLE: Covers the contents adequately but briefly; AUTHOR: May include author's name, or department and the like; PLACE OF PUBLICATION; DATE OF PUBLICATION; PUBLISHER; SERIAL NUMBER OR VOLUME NUMBER. PRICE, IF ANY. On the back of title page such information as: number and date of a revision or printing, the name of the printer and copyright notice may be set here.
***dedication**	This page if included follows immediately after the title page.
***foreword**	A general introduction, discussing the field covered and stressing the value of the work and the standing of the author. In governmental publications its place is sometimes replaced by a formal letter of transmittal from a minister or senior official.
***preface**	The preface is similar to the foreword but it is reserved for the author's introduction to his own publication; it may consist of:

In case of conflict between any statement in any article and the standard, the standard governs.

	A brief statement of the purpose of the publication.
	A general summary of the method of presenting the material;
	An acknowledgement of assistance from other individuals, departments or organizations.
	This represents the first page that is numbered, but not necessarily the first counted page; its page number starts in roman numerals.
***toc**	Table of contents: Defined as being a brief outline of the content of the book; it omits any reference to pages up to and including the table of contents.
***illus**	Illustrations: The style of this page usually follows the table of contents and also follows in the sequence to the toc.
***intro**	Introduction: In government publications, material that would otherwise be included in a preface sometimes appear as the first few paragraphs of the text.
***body**	Main portion of a document or publication which contains main text matter, divided into chapters with their headings and paragraphs.
***appendix**	Identifies major elements of a document which contains explanatory matter and illustrative material which is helpful to the reader but not essential to main text.
***schedule**	Identifies major elements of a document which contains explanatory matter and illustrative material which is helpful to the reader but not essential to main text (same as appendix).
***references**	Identifies the sources used to publish the book.
***abbreviations**	List of abbreviations used.
***glossary**	Partial dictionary of words or terms used in the book.
***bibliography**	The history or description of books and manuscripts, with notices of the edition, the dates of printing, etc.
***index**	Alphabetical list of parts with pages number.
***colophon**	Emblem of the publisher.

B. Code Definition or Description Parts of Text - Chapters and Headings

***h1**	Main head 1st level generally called a 'chapter'.
***h2**	Identifies a second-level heading or beginning of a segment, which identifies what the 1st level head is about.
***h3**	Begins and identifies 3rd level of which is generally called a part.
***h4**	Would identify a major head if there are no parts this would then replace a part.
***h5**	Fifth-level heading identities a part heading or next level to a major heading.
***h6**	Sixth level heading.
***hn**	nth level heading.

C. Code Definition or Description Parts of Text - Paragraphs

1. Main Paragraphs

Paragraphs are divided into hierarchy of importance, from the most important or main paragraph text to lower levels of sub-subdivisions.

***p**	This is an indicator of a main paragraph in the hierarchy of paragraphs.
***p1**	1st Level - This type of text or paragraph is somewhat similar to the

In case of conflict between any statement in any article and the standard, the standard governs.

	main paragraph; it would generally introduce a 2nd level paragraph.
***p2**	2nd Level - This type of text or paragraph is somewhat similar to the main and is generally introduced by the 1st level paragraph.
***p3**	3rd Level - Would also be similar to 1st and 2nd levels but would be on a lower level.
***p4, *p5, etc**	Would be still further levels of paragraphs and similar in fashion to previous levels.

2. Sub-Paragraphs

Paragraphs in the hierarchy are broken down into sub-paragraphs.

***ps**	This would represent a first part of a main level paragraph called a sub-paragraph; it differs somewhat from main level paragraphs in its make-up, also it complements its main paragraph.
***p1s**	This is the same type of paragraph only it complements its 1st level paragraph.
***p2s, *p3s, etc**	They are the same type but to lower levels of main.

3. Division-Paragraphs

Sub-paragraphs are also further divided into what could be called 'DIVISION-PARAGRAPHS' and these are identified by further numbering of sub-paragraphs.

***ps1**	This is a paragraph 1st level 'Division-Paragraph' and this is further divided into 'Sub-Division-paragraph'.
***ps2**	This would identify a main-paragraph sub-paragraph 'sub-division-paragraph'.
***ps3, *ps4, etc**	Would identify still further (sub)-division-paragraphs.

4. Quoted Paragraphs

These paragraphs would have the same identifiers as main (**p**, **p1**, **p2**, **p3** etc) and same as sub-paragraphs (**ps**, **p1s**, **p2s**, etc) and same as division-paragraphs (**ps1**, **ps2**, **ps3**, etc) but in addition would have the letter '**q**' added to the end of their identifier.

***pq**	Identifies a main level paragraph that is quoted.
***psq**	Identify main level sub-paragraph quoted.
***ps1q**	Identify 1st level division-paragraph quoted.

Paragraphs considered as quoted are text or phrases taken from other publications, journals, spoken by other persons other than the author or author's own works now being referred to.

5. Definition or Term Paragraphs

Another set of paragraphs needs to be identified, these are definition or term paragraphs, where a statement is given and therefore there are terms or definitions associated with this statement. Following are identifiers using the same hierarchy as main paragraphs and sub-paragraphs or division-paragraphs.

***pd**	Identifies a definition, description or explanation of a phrase or statement; this would be considered as being main level of definition.
***pd1**	This is the same level as a **p1**, only it would apply in a definition list category.
***pd2, pd3, etc**	Same level as **p2**, **p3**, etc of definition paragraphs.
***pds**	This is the same level as a sub-paragraph, only it would apply in a definition list category.
***pds1**	Also name category as **ps1** which is a division-paragraph 1st level paragraph now in a definition category.

6. Continuation Paragraphs

To be able to identify a paragraph after its interruption either by a level two paragraph, sub-paragraph or whatever this category of paragraph is called a continuation paragraph. These are indicated in the same fashion as all the other paragraphs with the addition of a '**c**' at the end.

In case of conflict between any statement in any article and the standard, the standard governs.

***pc**	Continuation main.
***p1c**	First level continuation.
***p2c**	2nd level continuation.
***psc**	Main level sub-paragraph continuation etc, etc
***pqc**	Main quoted continuation.

7. Extracted Paragraphs

These paragraphs would have the same identifiers as mains (**p**, **p1**, **p2**, **p3**, etc) and same as sub-paragraphs (**ps**, **p1s**, **p2s**, etc), also same as division-paragraphs (**ps1**, **ps2**, **ps3**, etc) but in addition would have the letter '**x**' added to the end of their identifier.

***px**	Main paragraph of an extraction.
***px1**	This is an indicator of a first level paragraph in an extraction category.
***px2**	2nd level extraction paragraph.
***psx**	Main sub-paragraph of an extraction.
***ps1x**	Main division-paragraph extracted.

Paragraphs considered as extraction are text or phrases taken from other publications, journals, etc. being referred to.

The problem with the extracted paragraph is that one can have extractions from different publications and in turn end up with two or more sets of paragraphs that have the same generic code but requiring different set-ups. Here we could add a letter or number to the '**x**' as a text identifier.

D. Code Definition or Description Other Features Commonly Used

***adrs**	First line of an address.
***adrs1**	Second line of an address.
***adrs3, adrs4, etc**	Other lines of an address, as needed.
***date**	Indicates a date line.

To be able to reference in the text where a footnote would follow a footnote reference is needed the following would suffice.

***fnref**	This would refer to a footnote.
***fnp**	Indicates the main paragraph of a footnote.
***fnp1**	Indicates the 1st level of a footnote.
***fnps**	Indicates a sub-paragraph of a main paragraph footnote.

The identifications would have the same levels as other paragraphs only with the addition of fn(number) to indicate to which footnote it belongs.

***line**	Any line that is out of the ordinary (a stand alone line).
***note**	Signifies a note but not a footnote.
***sep**	Separator is an indication that some characters are needed here to indicate the end of a subject and the beginning of another at this point.
***sg**	1st line of signature.
***sgs**	Signature sub-paragraph.
***sgs1**	Signature division-paragraph.
***sn**	Signifies a sidenote paragraph.
***time**	Indicates a time line.

To be able to distinguish certain words, phrases or group of phrases that are out of ordinary or to be emphasized.

***hl**	Beginning of highlighting.
***ehl**	To end highlighting.

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

Full-Text Data Base Publishing

Donald H. Rollert

Manager
Graphic Systems Development Division
U.S. Government Printing Office

Material originally presented at the
GCA 1983 Spring Conference
Philadelphia, 1983 May 12

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the GCA copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Graphic Communications Association. To copy otherwise, or to republish, requires a fee and specific permission. © 1983 GCA

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

I. INTRODUCTION

The Government Printing Office has converted from hot metal to photocomposition. This was a large undertaking and took 7 years to accomplish. Presently, GPO composes approximately 3 million pages a year in over 3,000 different page formats. Much more than a mere conversion from one process to another has been accomplished. The product line at GPO has been expanded. The actual data base, as well as the printed version of the publication, is now being sold. Because the system was designed to accept logically structured input, government agencies and firms in the private sector are presently utilizing the *full-text data base* in their information retrieval systems.

I do not want to bore you with a lot of technical data, but the concept behind the design and structuring of the *full-text data base* is so basic and important that I will spend most of the time available discussing the input for this system.

Computers were applied to typesetting by the major computer manufacturers in the early sixties. Unfortunately, they treated the typesetter as an application, rather than as a peripheral device. Input specifications developed were process-(display-) oriented, and any thought of the integrated data base and its use was ignored. For example, a catalog or price list could be marked up in accordance with the rules for composing copy used by the average newspaper today, and the data keyboarded at one of their text/edit terminals. The data would then be in a machine-recognizable form and could serve as input to their composition system, which would produce galleys of type in the specified format. However, this data (although in machine-recognizable form) can not qualify as a *full-text data base* because the display-oriented typesetting developments that are taking place in the computer industry, it is difficult and uneconomical, if not impossible, to develop computer systems that can use the "associative process" so natural to the human to sort out and analyze data. Thus, it is necessary when creating a computerized data base, to define this structure in a non-ambiguous form and to tag each element of like level with a *specified code that can be interpreted by the computer to mean one and only one thing*.

It is quite a paradox that, as the computer manufacturers were attacking the typesetting market (primarily the newspapers) as an application, problems then existing at GPO forced the management to direct its attention toward the data processing applications. At that time (early sixties), a large volume of computer printout was being delivered to GPO as camera-ready copy for publications having relatively short print runs. Although this printout was photo-reduced during the platemaking process, the product was still wasteful of paper, and the quality was very poor. The LINOTRON 1010 SYSTEM—including the Photocomposer and the MASTER TYPOGRAPHY PROGRAM—was delivered in 1967. Input to this system was magnetic tape submitted by the agencies. The agencies created these tapes by extracting data (primarily tabular) that they desired to publish from existing data systems. Obviously, these tapes contained no codes unique to the typesetting process, but the data was fielded or logically structured. GPO, during the following 4 or 5 years, developed the "edit-insert" idea which is based on the assumption that the typographic format can be determined by analyzing the structure of the data sent to GPO by the agency.

In 1975, when GPO started the conversion effort to photocomposition, the publications then being produced conventionally on hot metal equipment were much more complex than those being produced on the LINOTRON 1010 SYSTEM. The composition software then resident on GPO computers could not process the mixture of text, tables, and illustrations contained in these publications. High volume and schedule requirements dictated that a totally automated system that produced fully made-up pages of output be implemented. Although an extensive search of the marketplace for a system that could meet these requirements was made, none was found, and the decision was made to develop the software "in-house".

Experience with the LINOTRON SYSTEM made GPO realize the fundamental importance of developing a logically structured data base to serve as input to the system.

II. DATA BASE ANALYSIS

In developing the data base specifications, the various classes of data that could be encountered were analyzed. At first thought of a *full-text data base,* one thinks of only textual material—tables, illustrations and mathematical expressions are often not considered. The fact that only a small percentage of the data base may be tabular or mathematical in nature does not lessen the importance of this information. A *full-text data base* system must be designed to contain all types of information that can be expected to occur in the data base. The classes of data being considered by GPO are:

A. Text

Text is defined as the main body of written matter on the page. It includes, but is not limited to, paragraphs, fixed point leadering, (for example, a table of contents.), headings, and telephone leadering. This is a format in which left data is quadded left and right data is quadded right. The left and right data are usually, but not always, separated by leader codes. The name, address, and phone number entry in a phone book is text of a telephone leadering nature.

B. Tables

A table is defined as a condensed enumeration. The data is presented in a matrix form—i.e., a rectangular arrangement of data elements in rows and columns. In a table, the data displayed is not only condensed, but the reader is able

In case of conflict between any statement in any article and the standard, the standard governs.

to comprehend the information at a higher rate of speed than if the same information were presented in a textual format.

C. Symbolic

Mathematical and chemical expressions are a shorthand notation which can provide information at a glance, which would be very space-consuming and difficult to comprehend if expressed in a textual fashion.

D. Charts

The same data can be expressed in a number of ways, depending upon the point the author is trying to make. A graph or bar chart, for example, can indicate a trend better than a table. A table, however, presents detailed data that the bar chart cannot do.

E. Diagrams

Flow charts, pneumatic, hydraulic, and electrical circuits are by their nature very logical. Eventually, the GPO data base will be expanded to incorporate this class of data.

F. Mechanical Parts and Assemblies

Some work has been accomplished by industry in classifying and generating descriptions of mechanical parts in machine-recognizable form. Incorporating this class of data into the data base is much more complex than any of the others mentioned thus far. GPO at present has no plans to attack this problem.

G. Illustrations

Matter such as continuous tone photographs and line drawings within the present state-of-the-art cannot be resident in the data base in structured form. There are systems which digitize illustrations by scanning the original art, but this data is structured only for facsimile reproduction of the original art, and cannot be considered to be a data base in any sense of the word. Since the graphic's caption and legend describe the illustration, it is important that they, at least, be stored in the *full-text data base*. Even though size information is display-oriented, it is still necessary to store it in the data base.

Because of the nature of the publications incorporated into the system, GPO has concentrated to date on text, tables, and reservation of space for illustrations. Although the system became operational in 1976, new capabilities are constantly being added. For example, composition of mathematical expressions is under development. The most important task in this subsystem is the design of logically structured input for these equations.

Since a logically structured *full-text data base* has such great utility in many processes besides typesetting, one might think that it would cost more than display-oriented input to establish and maintain. Fortunately, this is not the case. It is much more time consuming and costly to prepare input for a display-oriented process, such as one of the computer-assisted galley composition programs in common use today, than for an automated composition system that accepts logically structured data. For straight text, the cost differential between the two approaches is not great. As the formats become more complex, however, data preparation costs for the logically structured approach become dramatically lower. Comparison of the two concepts can be made by examining the copy preparation and keyboarding required to compose the table shown in Example 1. Figure 1 shows the copy preparation required for the display-oriented method. Copy preparation marks for the line rules and boxhead especially are extensive and require a lot of painstaking work. The printout of the keyboarded input is shown in Figure 2. It is packed with function codes unique to the typesetting process. Both the printout of the input and the copy prepared manuscript for the logically structured method are shown in Figure 3. It took *five times* as long to prepare copy and keyboard the data for this rather simple table using the display-oriented method as opposed to the logically structured approach. Since the software for the automated system is responsible for the formatting of this table, copy preparation and input keyboarding are greatly simplified. This is especially true for the line rules and boxheads. On the other hand, most display-oriented composition system put much of the burden of formatting on the copy preparation and keyboarding, rather than building it into the composition program.

H. Sources

Certain data base material such as the *Congressional Record* and daily *Federal Register* are created "in-house". Most data bases, however, are purchased by GPO from private contractors. These contractors provide reliable service—quick turnaround at low prices. GPO also composes a large number of publications from data produced by a number of government agencies to GPO specifications.

The system is designed and the procedures are set up so that data for a given publication can come from a number of sources. For example, some of the bills, committee reports, and roll call material contained in the *Congressional Record* comes directly to GPO on magnetic tape, while the remaining material is sent in manuscript form to be keyed by GPO. Gradually, more data will come in machine-recognizable form, and less in manuscript form.

GPO is making a concerted effort to encourage customers to send material to GPO in machine-recognizable form as opposed to manuscript form. To be viable—i.e., to save time and money—the data must be structured logically and

In case of conflict between any statement in any article and the standard, the standard governs.

verified. The initial keyboarding is only the first step in the creation of the data base, and accounts for less than 25 percent of the total cost. The major costs are incurred in the proofing and correcting phase.

Experience has shown that it is more economical for GPO to create a data base from manuscript, rather than try to use material from a non-verified or process-oriented data tape.

Data bases can be economically created to GPO specifications on many types of equipment, from paper tape perforators to the most elaborate text editing or word processing systems.

I. Word Processing

With the proliferation of word processing equipment throughout the agencies, why is so much of the input still coming to GPO in manuscript form? Word processing systems have made it possible for the typist to produce and especially to update a document much easier and quicker. Quality of the product is also higher. The basic responsibilities of the typist, however, have not changed. The computer equipment associated with word processing is merely used to assist the word processor operator to perform basic typing functions easier and more economically. For most applications to which the word processors are being applied, this basic approach—assisting the operator to be more accurate and efficient—is very valid.

If word processing equipment is to play a meaningful part in the creation of structured data bases, two things must happen:

(1) The word processor operator must key the data from the manuscript in accordance with the specifications for the structured data base.

(2) The display on the terminal and/or the printout from the printer associated with the word processor cluster of terminals must provide a formatted presentation that is easy to read and comprehend.

In most word processing applications, a draft of the material in readable form is needed within minutes after keying if the organization using the system is to function efficiently. Most printer software that is presently available on word processing equipment is very crude, and the printout of logically structured input for a table would look more like the input data shown in Figure 3 than the composed presentation of the table. Needless to say, this would not be very useful to the reader. Since a readable draft of the table is needed immediately, the word processor operator is required to set up tab points and to properly "space out" the data so that a presentable table appears on the screen, and will be printed out on the printer. Although it takes the word processor operator longer to key in the display-oriented as opposed to the structured mode, the tables produced will, in all probability, not be as well designed and proportioned. Unfortunately, present day printer and display software limitations for word processing equipment dictate this approach.

Again it is emphasized that the "spaced out" data keyed by the operator, although in machine-recognizable form, cannot be regarded as a data base. Some manufacturers have attempted to write computer software that will edit this display-oriented data into a logically structured data base. Except for straight text, these efforts have not been successful, and almost no data base of any consequence is made up of only straight text.

The best approach is to develop computer logic that can produce displays and printouts in composed form from logically structured data. To make this a reality will be no simple task that will happen overnight, but there are developments occurring that make this concept a very practical one. The quality and versatility of display terminals and printers are improving markedly while their cost is dropping. Complex computer software is no longer restricted to the main frame or minicomputer.

The Offices of Legal Counsel for both the Senate and House keyboard bills in structured form to GPO specifications. GPO has provided them with a computer program that can accept this logically structured data as input and produce a fairly good looking draft on their line printers. Tables as well as text can be processed by this printer program, but the boxheads produced are less than satisfactory.

J. Potential

An impact printer with a limited character complement has serious mechanical limitations for printing drafts of legal or technical material. However, nonimpact printers are being developed that produce output resolution of 200 or more pixels per inch. Although these resolutions do not approach those of the phototypesetter, they are high enough to utilize the tools of the graphics designer for formatting. Very attractive formats of complex material can be produced on this class of printer. Much of the software development for fully automated composition on phototypesetters can be applied to the driving of this class of printer from logically structured input.

The price of these printers will have to come down, and their reliability must be improved, before there can be wide application for use with word processing systems. Although required computer resources (central processing unit and memory) are already available at reasonable prices, the necessary software for composition by the printers from the structured input does not exist. Fortunately, what may be a problem to the user is often regarded as an opportunity by a manufacturer.

Design and implementation of publication systems is the primary function of the Graphic Systems Development Di-

In case of conflict between any statement in any article and the standard, the standard governs.

vision. The Electronic Systems Development Division is responsible for keeping abreast of the hardware developments and can provide assistance to the agency in specification and selection of hardware. These capabilities together with the production resources housed in the Electronic Photocomposition Division make it possible for GPO to start an agency with a major publishing problem on a path similar to the ones that the following publications have taken: *U.S. Code, Daily Federal Register, Code of Federal Regulations, Decisions and Orders of the National Labor Relations Board, Statistical Abstract of the United States,* and the *Congressional Record.* Initially, production personnel at GPO play a very important part in getting the system off the ground. Gradually, most of the direct labor functions are absorbed by agency personnel which gives them better control over costs and schedules. Most important of all, however, by establishing and using a current structured data base, the publication system can take advantage of future technological developments. So many ideas and products are emerging that it is impossible to know which will succeed and which will fall by the wayside. We do know, however, that a logically structured data base is and will be the foundation upon which to build, and display of information for reading by a human will be an important feature in any system.

Before ending this discussion I want to describe an experiment which was conducted. Does such a system as this have application in the preparation of technical manuals and scientific journals? Although scientific journals and technical manuals have much in common with the legal publications which the AUTOMATED COMPOSITION SYSTEM was designed to produce, there are basic differences. The legal publication places primary emphasis on text, tables, footnotes, and continued heads. While scientific journals also utilize footnotes and tables, greater importance is placed on the ability to compose mathematical and chemical equations and expressions. Technical manuals are usually very heavily illustrated. At first glance, it might seem that the text is inserted as captions and legends into the illustrations which make up the bulk of the publication. This, however, is not the case. It is important to understand that the text is the "skeleton" which shapes and organizes the publication.

Presently, many line drawings are stored in machine-recognizable form. Although great progress is being made in the fields of computer-assisted design and interactive computer graphics, most of this data is not logically structured and cannot be considered a data base. However, an automated system must have the capability of electronically "stripping in" these illustrations as it writes the driver for the photocomposer or nonimpact printer. Positioning the graphic relative to the text, however, is the major problem. Controlling the graphic scanning is, by comparison, a trivial task.

If an automated system based on the principles of the GPO system is to be used in an economic manner to produce complex technical manuals, it must comply with the following requirements:

(1) Able to accept input data that is in a logically structured form;

(2) Automatically produce complete pages containing all the classes of data expected to appear in the manual. Conform to the page format design contained in the specifications;

(3) Have a turnaround time for the composition of a chapter or section of less than 10 minutes. In other words, the system must appear to the writer like an intelligent copier;

(4) The equipment configuration must be such that the cost is no more than twenty-five cents per page per iteration;

(5) Be capable of building the driver for the specified phototypesetter during the final pass.

Publishing From a Full Text Data Base[1] describes the system now in use by GPO. Since this manual has formatting requirements very similar to those required for software documentation, it was decided to simulate such a system by using the AUTOMATED COMPOSITION SYSTEM to compose this publication. Maybe this effort will provide at least a partial answer to this question.

To economically justify a printer configuration such as the Comp80/2 and associated microfiche copier or the Xerox 9700, the operation would have to produce at least 600 thousand final pages per year. It would also be difficult to provide the 10- to 15-minute turnaround time (necessary if the author is to work efficiently) from a large centrally located page printer.

Hopefully, lower cost nonimpact printers (under $20,000) will become available within the next few years. Until they do, the use of automated phototypesetting systems for this application will be limited to a few high-volume producers.

Present systems are resident on either main frames or minicomputers. To operate in conjunction with the low-cost printer, the composition system will have to be resident on either the text/edit (word processing) system or microcomputer associated with the printer itself. Technologically and economically, both of these approaches are very feasible.

The AUTOMATED COMPOSITION SYSTEM, although quite sophisticated, would have to be greatly enhanced if it were to be applied to a very broad spectrum of technical publications.

As far as the GPO system is concerned, the following enhancements would broaden the spectrum of economic application for the system in technical manual and scientific journal publishing:

In case of conflict between any statement in any article and the standard, the standard governs.

(1) Compose complex mathematical and chemical expressions and equations from logically structured input. Often these expressions must be composed within a line of text or in the cell of a table. Some multti-level equations cannot fit into the column width. It is essential that the system automatically break an equation of this type into a series of lines;

(2) Integrate text and graphics, to the extent that line art stored in machine-recognizable form can be electronically "stripped in" during the composition pass;

(3) Integrate computer graphics so that, on demand, specified data in the data base could be presented in chart, as opposed to tabular form. Fortunately, the developers of computer graphics have paid much more attention to logically structured input than have the developers of composition systems;

(4) Routines which provide for the floating of tables, diagrams, and illustrations must be improved. Floated material should be placed at the bottom of columns as well as at the top;

(5) Provide for genuine integration of graphics as the state of the art of computer-assisted design advances. The potential of these *design data bases* is as great as that of the *full-text data base;*

(6) Compute word space values for line justification based on analysis of an entire paragraph as opposed to the single line being constructed;

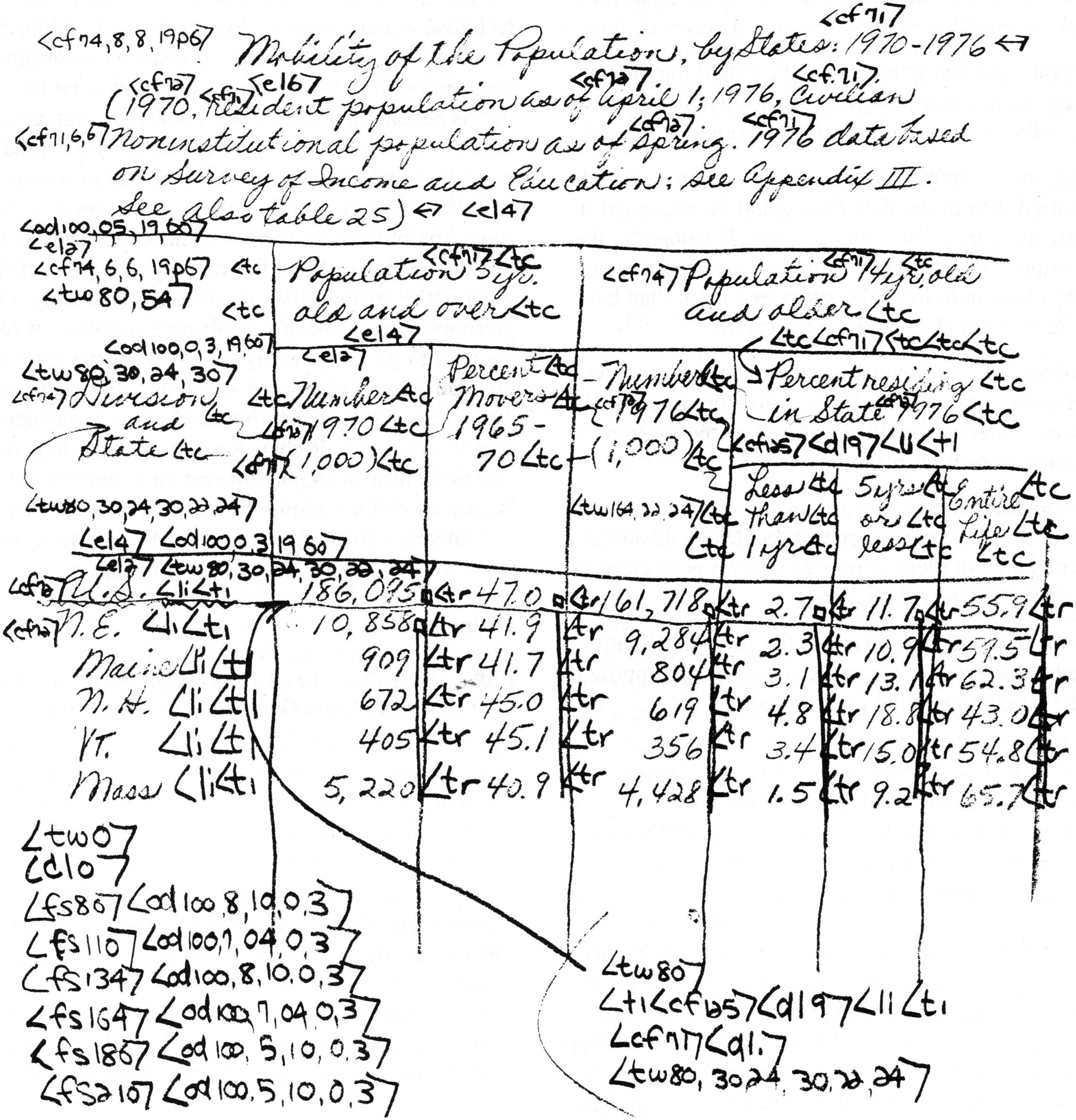
(7) Arrange pages based on an analysis of a group of pages, to determine page dimensions that will produce the lowest number of unjustified columns.

In conclusion, Figure 4 is a diagram that sums up what we are trying to do whether it be in legal publishing or technical documentation. Especially with publications like the *U.S. Code* and *BILLS,* a system conforming to this diagram (which, by the way, looks somewhat like a logo to me) is becoming a reality. We know that a lot remains to be done, but we believe that we are on the right road and that we will improve service and save money as we travel it. We have had experience during the past several years that should be of value to various agencies in both the Congress and the Executive Branch. So many new products are coming on the market that it must be confusing to agency personnel with a specific publishing problem. It seems that we tend to go off in many directions at the same time and this can waste large sums of money and cause service to deteriorate. GPO desires not to be one more hurdle for a user to overcome. Rather, we want to provide good service and assistance in the automation of a publication process. We are careful not to misapply our system. If we know of a system that can perform the specific function better than ours, we will say so and will recommend it.

Endnotes

1. *Publishing From A Full Text Data Base,* 3d ed. (Washington, D.C.: United States Government Printing Office, 1983).

In case of conflict between any statement in any article and the standard, the standard governs.



<cf74,8,8,19p6> Mobility of the Population, by States: 1970-1976 <cf71> <e>
(1970, <cf72> <cf71> <el6> resident population as of April 1; <cf72> 1976, <cf.71> civilian
<cf71,6,6> noninstitutional population as of spring. <cf72> 1976 <cf71> data based
on Survey of Income and Education; see Appendix III.
See also table 25) <e> <el4>

<od100,05,19,60>

<el2> <cf74,6,6,19p6> <tc <tw80,54> <tc	<cf71> <tc Population 5 yr. old and over <tc <el4>		<cf74> Population <cf71> 14 yr. old and older <tc			
<od100,0,3,19,60> <tw80,30,24,30> <cf74> Division and State <tc <tw80,30,24,30,22,24>	<el2> Number <tc <cf72> 1970 <tc <cf71> (1,000) <tc	Percent <tc movers <tc 1965-70 <tc	Number <tc <cf72> 1976 <tc (1,000) <tc <tw64,22,24> <tc	<tc <cf71> <tc <tc <tc Percent residing <tc in State <cf72> 1976 <tc <cf65> <dl9> <ll <tl		
<el4> <od100,0,3,19,60> <el2> <tw80,30,24,30,22,24>				Less <tc than <tc 1 yr <tc <tc	5 yrs <tc or <tc less <tc	Entire <tc life <tc <tc
<cf72> U.S. <li <ti	186,095 □ <tr	47.0 □ <tr	161,718 □ <tr	2.7 □ <tr	11.7 □ <tr	55.9 <tr
<cf72> N.E. <li <ti	10,858 □ <tr	41.9 <tr	9,284 <tr	2.3 <tr	10.9 <tr	57.5 <tr
Maine <li <ti	909 <tr	41.7 <tr	804 <tr	3.1 <tr	13.1 <tr	62.3 <tr
N.H. <li <ti	672 <tr	45.0 <tr	619 <tr	4.8 <tr	18.8 <tr	43.0 <tr
Vt. <li <ti	405 <tr	45.1 <tr	356 <tr	3.4 <tr	15.0 <tr	54.8 <tr
Mass. <li <ti	5,220 <tr	40.9 <tr	4,428 <tr	1.5 <tr	9.2 <tr	65.7 <tr

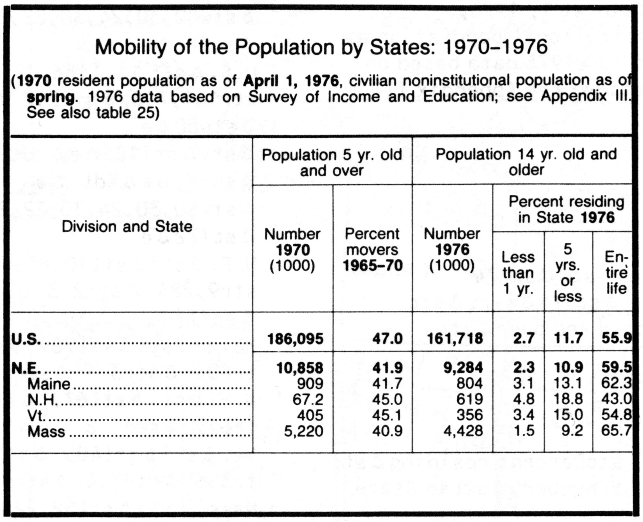
<tw0>
<dl0>
<fs86> <od100,8,10,0,3>
<fs110> <od100,1,04,0,3>
<fs134> <od100,8,10,0,3>
<fs164> <od100,7,04,0,3>
<fs186> <od100,5,10,0,3>
<fs210> <od100,5,10,0,3>

<tw80>
<ti <cf65> <dl9> <ll <ti
<cf71> <dl.>
<tw80,30,24,30,22,24>

FIGURE 1.
Display-Oriented
Manuscript

```
Δscf74,8,8,19p6Δe
Mobility of the Population, by States:
Δscf71Δe1970-1976Δc
Δsel6Δe
Δsvb.20Δe
Δscf71,6,6Δe(Δscf72Δe1970, Δscf71Δeresident
population as of Δscf72ΔeApril 1; 1976,
Δscf71Δecivilian noninstitutional population as
of Δscf72Δespring. Δscf71Δe1976 data based on
Survey of Income and Education; see Appendix III.
See also table 25)Δc
Δsel4Δe
Δsod100,0,5,19,60Δe
Δsel2Δe
Δscf74,6,6,19p6Δe
Δstw80,54Δe
ΔstcPopulation Δscf71Δe5ΔstcΔscf74
ΔePopulation Δscf71Δe14 Δscf74Δeyr.Δstc
Δstcyr. old and overΔstcold and overΔstc
Δsel4Δe
Δsod100,0,3,19,60Δe
Δsel2Δe
Δstw80,30,24,30Δe
ΔstcΔscf71ΔeΔstcΔstcΔstcPercent residingΔstc
ΔstcNumberΔstcPercentΔstcNumber,Δstcin State
Δscf72Δe1976Δstc
Δscf74ΔedivisionΔstcΔs
cf72Δe1970ΔstcΔscf71Δe
MoversΔstcΔscf72Δe1976ΔstcΔscf125ΔeΔsdl9Δe9
99Δsfs-3mΔeΔsliΔstl
Δscf71ΔeΔsdl.Δe
Δstw80,30,24,30,22,24Δe
Δscf74Δeand stateΔstcΔscf71Δe(1,000)Δs
tcΔscf72Δe1965-70ΔstcΔscf71Δe(1,000)Δs
tcLessΔstc5 yrs.ΔstcEntireΔstc
Δstw164,22,24Δe
ΔstcthanΔstcorΔstcLifeΔstc
Δstc1 yr.ΔstclessΔstcΔstc
Δsel4Δe
Δsod100,0,3,19,60Δe
Δsel2Δe
Δstw80,30,24,30,22,24Δe
Δscf72Δe
U.S.ΔsliΔstl186,095 Δstr47.0 Δ
str161,718 Δstr2.7 Δstr11.7 Δstr55.9Δstr
Δstw80Δe
ΔstlΔscf125ΔeΔsdl9Δe999Δsfs-3mΔeΔsliΔstl
Δscf71ΔeΔsdl.Δe
Δstw80,30,24,30,22,24Δe
Δscf72Δe
N.E.ΔsliΔstl10,858 Δstr41.9 Δ
str9,284 Δstr2.3 Δstr10.9 Δstr59.5Δstr
Δscf71Δe
MaineΔsliΔstl909 Δstr41.7 Δs
tr804 Δstr3.1 Δstr13.1 Δstr62.3Δstr
N.H.ΔsliΔstl672 Δstr45.0 Δs
tr619 Δstr4.8 Δstr18.8 Δstr43.0Δstr
Vt.ΔsliΔstl405 Δstr45.1 Δs
tr356 Δstr3.4 Δstr15.0 Δstr54.8Δstr
Mass.ΔsliΔstl52,220 Δstr40.9 Δ
str4,428 Δstr1.5 Δstr9.2 Δstr65.7Δstr
Δstw0Δe
Δscl0Δe
Δsfs80ΔeΔsod100,8,10,0,3Δe
Δsfs110ΔeΔsod100,7,04,0,3Δe
Δsfs134ΔeΔsod100,8,10,0,3Δe
Δsfs164ΔeΔsod100,7,04,0,3Δe
Δsfs186ΔeΔsod100,5,10,0,3Δe
Δsfs210ΔeΔsod100,5,10,0,3Δe
```

FIGURE 2.
Display-Oriented Method
Input



Mobility of the Population by States: 1970–1976

(**1970** resident population as of **April 1, 1976**, civilian noninstitutional population as of **spring**. 1976 data based on Survey of Income and Education; see Appendix III. See also table 25)

Division and State	Population 5 yr. old and over		Population 14 yr. old and older			
				Percent residing in State **1976**		
	Number **1970** (1000)	Percent movers **1965–70**	Number **1976** (1000)	Less than 1 yr.	5 yrs. or less	Entire life
U.S.	**186,095**	**47.0**	**161,718**	**2.7**	**11.7**	**55.9**
N.E.	**10,858**	**41.9**	**9,284**	**2.3**	**10.9**	**59.5**
Maine	909	41.7	804	3.1	13.1	62.3
N.H.	67.2	45.0	619	4.8	18.8	43.0
Vt.	405	45.1	356	3.4	15.0	54.8
Mass	5,220	40.9	4,428	1.5	9.2	65.7

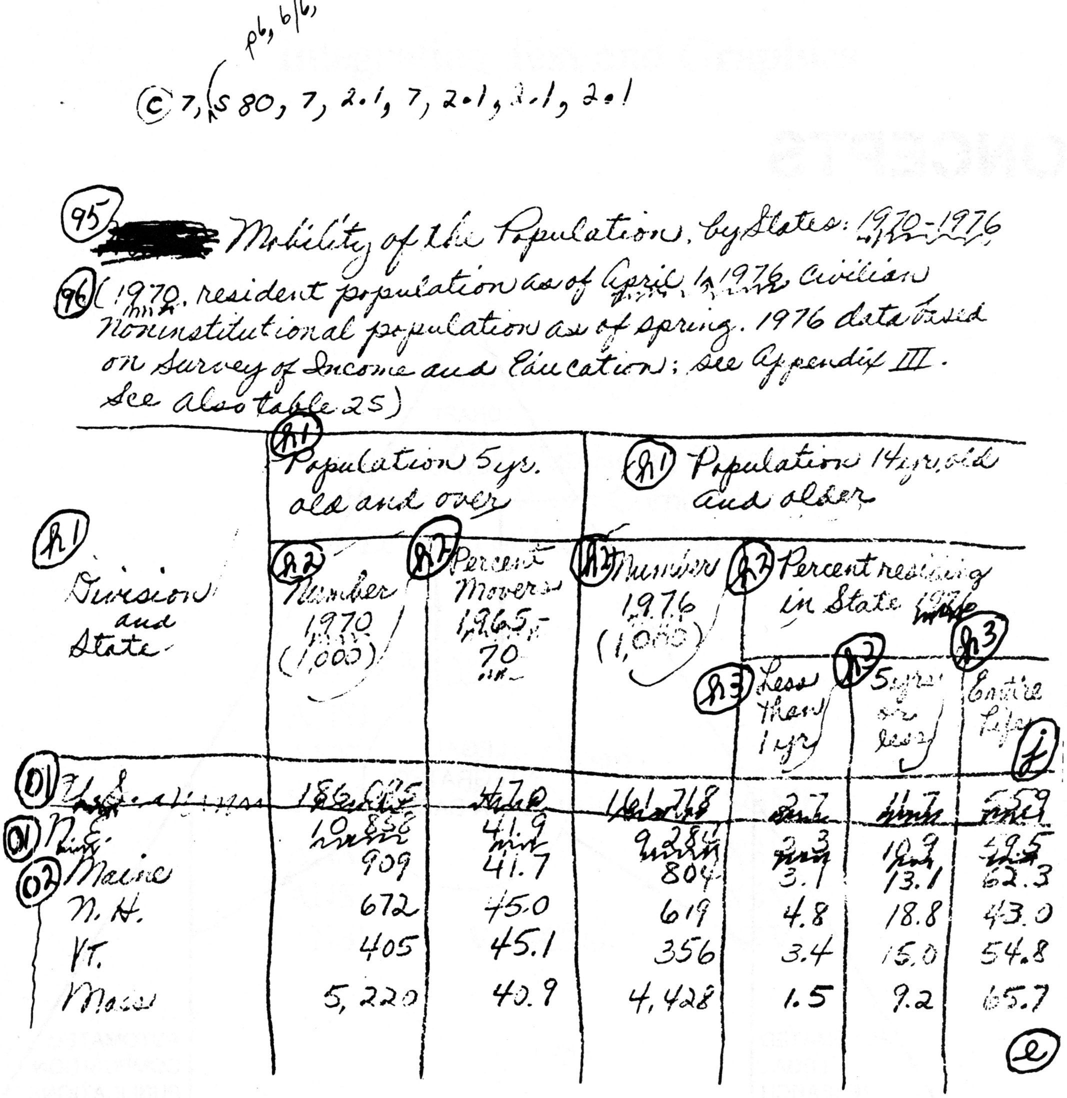
EXAMPLE 1.
Representative Table

Note

Example 1 is sequenced between Figure 2 and Figure 3 to allow ready comparison of the tables as set with two different methods of markup.

Chapter 1—The Full Text Data Base

Manuscript and Input



p6, 6/6,

Ⓒ 7, s80, 7, 2.1, 7, 2.1, 2.1, 2.1

(95) [redacted] Mobility of the Population, by States: 1970-1976

(96) (1970 resident population as of April 1, 1976 civilian noninstitutional population as of spring. 1976 data based on Survey of Income and Education; see Appendix III. See also table 25)

(h1) Division and State	(h1) Population 5 yr. old and over		(h1) Population 14 yr. old and older				
	(h2) Number 1970 (1,000)	(h2) Percent movers 1965-70	(h2) Number 1976 (1,000)	(h2) Percent residing in State 1976			
				(h3) Less than 1 yr	(h3) 5 yrs or less	(h3) Entire life	(j)
(01) U.S.	186,095	47.0	161,718	2.7	11.7	55.9	
(01) N.E.	10,858	41.9	9,284	2.3	10.9	59.5	
(02) Maine	909	41.7	804	3.1	13.1	62.3	
N.H.	672	45.0	619	4.8	18.8	43.0	
Vt.	405	45.1	356	3.4	15.0	54.8	
Mass	5,220	40.9	4,428	1.5	9.2	65.7	

(e)

Input Manuscript (Logically Structured)

```
∆c7,p6,6/6,s80,7,2.1,7,2.1,2.1,2.1
∆I95Mobility of the Population by States: 1970-
1976
∆I96(∆T21970∆T1 resident population as of
∆T2April 1, 1976∆T1, civilian noninstitutional
population as of ∆T2spring∆T1. 1976 data based on
Survey of Income and Education; see Appendix III.
See also table 25)
∆h1Division and State
∆h1Population 5 yr. old and over
∆h2Number ∆T21970∆T1 (1000)
∆h2Percent movers ∆T21965-70∆T1
∆h1Population 14 yr. old and older
∆h2Number ∆T21976∆T1 (1000)
∆h2Percent residing in State ∆T21976∆T1
∆h3Less than 1 yr.
∆h35 yrs. or less
∆h3Entire life
∆j
∆I01∆T2U.S.∆D∆T2186,095∆D∆T247.0∆D
∆T2161,718∆D∆T22.7∆D∆T211.7∆D∆T255.9∆rn,s→
∆I01∆T2N.E.∆D∆T210,858∆D∆T241.9∆D
∆T29,284∆D∆T22.3∆D∆T210.9∆D∆T259.5
∆I02Maine∆D909∆D41.7∆D804∆D3.1∆D13.1∆D62.3
∆I02N.H.∆D67.2∆D45.0∆D619∆D4.8∆D18.8∆D43.0
∆I02Vt.∆D405∆D45.1∆D356∆D3.4∆D15.0∆D54.8
∆I02Mass∆D5,220∆D40.9∆D4,428∆D1.5∆D9.2∆D65.7
∆e
```

FIGURE 3.
Logically Structured Method

CONCEPTS

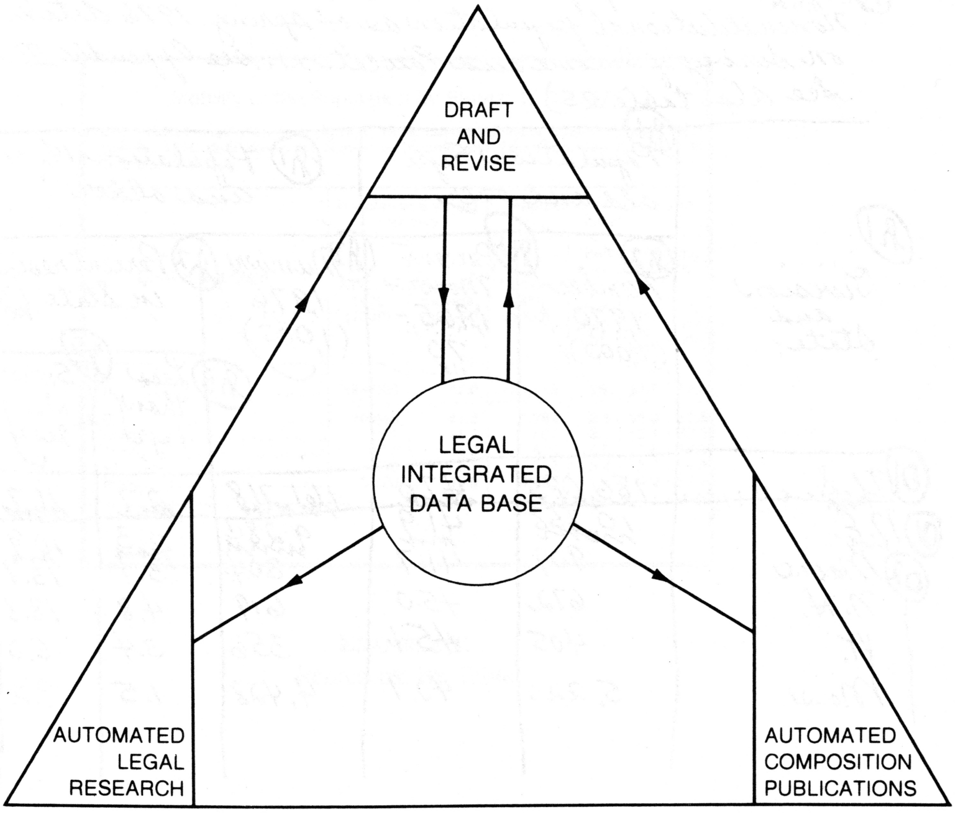

DRAFT AND REVISE
LEGAL INTEGRATED DATA BASE
AUTOMATED LEGAL RESEARCH
AUTOMATED COMPOSITION PUBLICATIONS

FIGURE 4.
Flow Diagram of the Legal Integrated Data Base

Integrating Text and Graphics

Aaron H. Bigman

Senior Financial Publishing Consultant
Standard & Poor's Corporation
(A Division of McGraw-Hill, Inc.)

Former Member

ANSI		X3J6
ANSI	Task Group	X3J6.1
EG CLPT	Task Group	SGML
ANSI		X3V1
ANSI		X3V1.2
ANSI	Vice-Chair	X3V1.3

1982

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the GCA copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Graphic Communications Association. To copy otherwise, or to republish, requires a fee and specific permission. © 1983 GCA

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

A. Introduction

The Graphic Communications Association (GCA) surveyed technical publication and production managers of large aerospace and heavy equipment manufacturers to determine the extent to which these industries had automated their publication production. GCA selected this industrial community because it had been actively automating information processing functions and integrating them into the production of technical documents.

The term "technical documentation" refers to the maintenance, operating, and training manuals required by complex and sophisticated military and industrial equipment and projects. This definition also includes catalogs and illustrated parts breakdown listings. Specifications dictating the appearance of these materials is established by the military or commercial user; traditionally this has meant using 8½ x 11-inch pages printed with black ink and a two-column text format that is interspersed with graphics ranging from schematics and flow charts to illustrations and complex engineering drawings.

B. Results Summary

Survey responses can be summarized in four major points:

(1) The majority of technical production facilities have automated the text preparation and presentation functions.

(2) Graphic automation ranging, again, from technical illustrations to complex engineering drawings is rapidly approaching this level.

(3) Among those production facilities that had automated both text and graphic preparation, only a handful had automated the interface for the integrated manufacturing of technical documentation. However, *all responses* cited that the merging process was currently being studied.

(4) The successful, automated integration of geometric and alphanumeric data would tremendously enhance the productivity and efficiency of technical documentation manufacturing.

C. System Integration

As demonstrated in the survey responses, text entry and editing is almost always computer assisted. Although compatibility issues continue to exist between various formatting and editing systems, word processors and terminals have automated text input. Computer graphic systems have proliferated throughout the engineering, manufacturing, and quality assurance areas of industrial manufacturing and are rapidly finding their way into the technical publication departments. In any event, chances are good that the artwork has been computer generated. At this point, however, progress toward the automated/integrated technical publication system starts to slow.

The main reason for the lack of progress in text and graphic interfacing is the problem of system integration. System integration in a technical publication environment is primarily the merging and formatting of alphanumeric and graphic (raster, vector, synthetic) for the production of a technical document page. The problem of system integration is really one of vertical integration—publishing being the most "vertical" of many information processing exercises. Vertical integration is building compatibility among standalone processing devices that each treat and manipulate data in different formats. Compatible systems would, therefore, be able to exchange and process data for the manufacturing of a product. Interfacing word and text processing and computer generated graphics for phototypesetting or electronic printing is an issue for phototypesetting or electronic printing is an issue of vertical integration.

System integration is obviously not a problem that is unique to automated technical documentation production. The Air Force is sponsoring an effort called ICAM (Integrated Computer Aided Manufacturing) to examine the role of computers in the manufacturing process. ICAM identified the issue of non-integrated systems as one of the five most critical problems in the effort to enhance manufacturing productivity through automation. (Technical documentation is, in fact, a manufacturing process as is weapon system and airplane production.) An ICAM conclusion was that no two independently developed systems can be integrated unless they have parts in common. Vertical system integration, therefore, resides in finding "commonalities" between text and graphic data structures for formatting and typesetting.

D. Current Approaches

Before the issue of common elements is explored, it is useful to examine current approaches to automated publication systems. The essential issue is that merging a text and a graphic data stream is not an end in itself, but part of the process toward pagination. Therefore, to produce made up pages of text and graphics, graphics are integrated *into* the text stream that contains the alphanumeric information and the format and presentation parameters. This "overhead" that has been mastered by text processing systtems is not readily transferrable to CAD/CAM devices. Therefore, pagination initially involves text and pictorial layout where it is not essential to reproduce the graphic element, but its "space" requirement on the page. The next step is to ensure that the output device can mark the placeholder for the insert and replicate the graphic image. Replication at this level is pictorial and two dimensional, with the assumption that all editing operations have been done on either the graphics or text system.

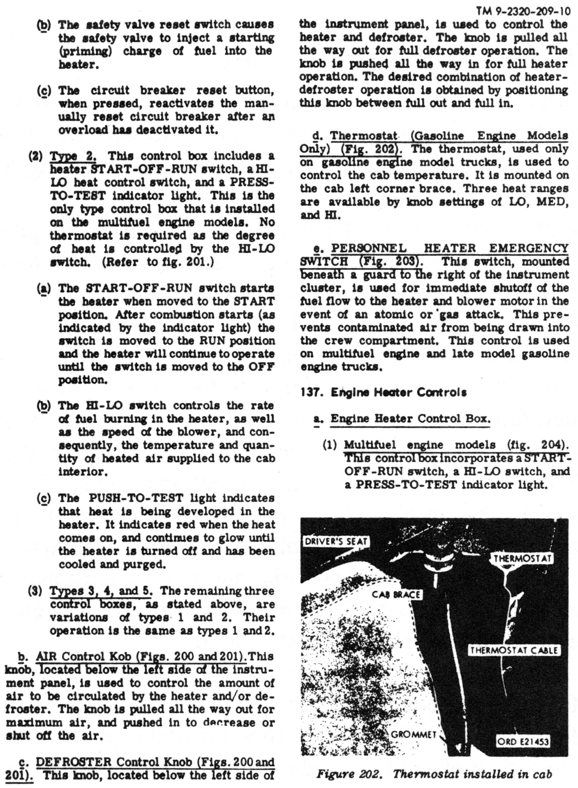
The character/symbol generators of the current generation of typesetting equipment are not conducive to the mastering of pictorial data. Typesetters function best in an environment where they are fed a steady stream of sequential, encoded data, and by accessing typographic font libraries, can read the instructional coding and expose the desired presentation of the data. The stroking features of CRT digital typesetters

In case of conflict between any statement in any article and the standard, the standard governs.

are thus oriented toward character generation and not toward straight lines, arcs, curves, polylines, angles, etc. Although any device that can image data (or ovals, lines) on a page can generate pictorial data, the amount of non-encoded instructional data that the output device needs to process degrades system throughput. Despite these issues, Information International, Inc., Autologic, and Mergenthaler have all demonstrated both graphic and halftone quality output on their current line of equipment by designing special graphic fonts. These fonts consist of fixed or variable width line segments or other graphic primitive elements. Fonts correspond to pen ink densities and cause the typesetter to function as a plotter. Both the III and Autologic systems are currently used for the generation of technical document pages (consisting of text and line art) in production environments. Work is currently underway to develop compression techniques for scanned and "rasterized" line segment data to speed up the processing times per page. The GCA survey cites this integrated text/graphics production as the key to an automated document manufacturing process, and thus a potential market will justify the time and money expenditures.

E. Graphics Emphasis

Why is graphics output suddenly so significant to technical document manufacturing? The reason lies outside the realm of printing systems technology. Technical documentation is, in essence, nothing more than the delivery vehicle for the distribution of technical information. This information has been "formatted" by a plethora of government and commercial specifications that govern the presentation of the information. This presentation, to date, has been a tradeoff between cost, aesthetics, and technology. Until most recently, the delivery style was narrative text, and thus the document producers geared up and automated their facilities to produce text characters as fast and as efficiently as possible. The *intent* of the information, however, is to facilitate the maintenance, repair or servicing of a system of some type. Therefore, the content of the information has to be legible, comprehensible, and self-explanatory at a number of various education and skill levels. The outside factor thus becomes the educational psychologists and readability experts who believe that the traditional delivery format is no longer effective. Comprehension can only be obtained by a more graphical orientation of the technical data, a method of presentation dramatically different from current publication practices (see Figures 1 and 2). This graphics orientation has, within the last year, started to affect the manufacturing of technical information. The Air Force Job Guides and the Army's New Look are ongoing efforts to re-format technical manuals and introduce graphics as the primary element in information delivery.



TM 9-2320-209-10

(b) The safety valve reset switch causes the safety valve to inject a starting (priming) charge of fuel into the heater.

(c) The circuit breaker reset button, when pressed, reactivates the manually reset circuit breaker after an overload has deactivated it.

(2) Type 2. This control box includes a heater START-OFF-RUN switch, a HI-LO heat control switch, and a PRESS-TO-TEST indicator light. This is the only type control box that is installed on the multifuel engine models. No thermostat is required as the degree of heat is controlled by the HI-LO switch. (Refer to fig. 201.)

(a) The START-OFF-RUN switch starts the heater when moved to the START position. After combustion starts (as indicated by the indicator light) the switch is moved to the RUN position and the heater will continue to operate until the switch is moved to the OFF position.

(b) The HI-LO switch controls the rate of fuel burning in the heater, as well as the speed of the blower, and consequently, the temperature and quantity of heated air supplied to the cab interior.

(c) The PUSH-TO-TEST light indicates that heat is being developed in the heater. It indicates red when the heat comes on, and continues to glow until the heater is turned off and has been cooled and purged.

(3) Types 3, 4, and 5. The remaining three control boxes, as stated above, are variations of types 1 and 2. Their operation is the same as types 1 and 2.

b. AIR Control Kob (Figs. 200 and 201). This knob, located below the left side of the instrument panel, is used to control the amount of air to be circulated by the heater and/or defroster. The knob is pulled all the way out for maximum air, and pushed in to decrease or shut off the air.

c. DEFROSTER Control Knob (Figs. 200 and 201). This knob, located below the left side of the instrument panel, is used to control the heater and defroster. The knob is pulled all the way out for full defroster operation. The knob is pushed all the way in for full heater operation. The desired combination of heater-defroster operation is obtained by positioning this knob between full out and full in.

d. Thermostat (Gasoline Engine Models Only) (Fig. 202). The thermostat, used only on gasoline engine model trucks, is used to control the cab temperature. It is mounted on the cab left corner brace. Three heat ranges are available by knob settings of LO, MED, and HI.

e. PERSONNEL HEATER EMERGENCY SWITCH (Fig. 203). This switch, mounted beneath a guard to the right of the instrument cluster, is used for immediate shutoff of the fuel flow to the heater and blower motor in the event of an atomic or gas attack. This prevents contaminated air from being drawn into the crew compartment. This control is used on multifuel engine and late model gasoline engine trucks.

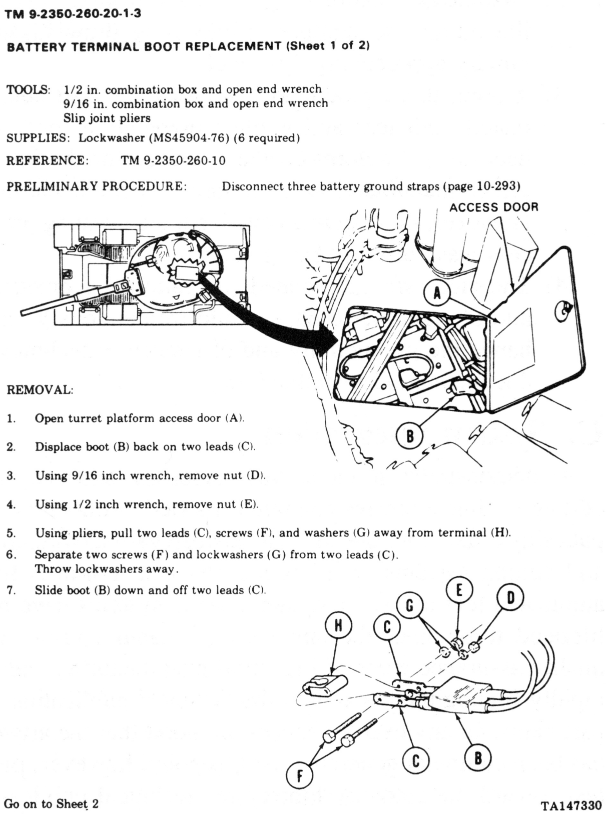
137. Engine Heater Controls

a. Engine Heater Control Box.

(1) Multifuel engine models (fig. 204). This control box incorporates a START-OFF-RUN switch, a HI-LO switch, and a PRESS-TO-TEST indicator light.

Figure 202. Thermostat installed in cab

Figure 1



TM 9-2350-260-20-1-3

BATTERY TERMINAL BOOT REPLACEMENT (Sheet 1 of 2)

TOOLS: 1/2 in. combination box and open end wrench
9/16 in. combination box and open end wrench
Slip joint pliers

SUPPLIES: Lockwasher (MS45904-76) (6 required)

REFERENCE: TM 9-2350-260-10

PRELIMINARY PROCEDURE: Disconnect three battery ground straps (page 10-293)

ACCESS DOOR
A
B

REMOVAL:

1. Open turret platform access door (A).
2. Displace boot (B) back on two leads (C).
3. Using 9/16 inch wrench, remove nut (D).
4. Using 1/2 inch wrench, remove nut (E).
5. Using pliers, pull two leads (C), screws (F), and washers (G) away from terminal (H).
6. Separate two screws (F) and lockwashers (G) from two leads (C).
Throw lockwashers away.
7. Slide boot (B) down and off two leads (C).

Go on to Sheet 2

TA147330

Figure 2

In case of conflict between any statement in any article and the standard, the standard governs.

The introduction of graphic-oriented page layouts should pose a severe problem for the current manufacturers of technical information. These facilities are oriented toward text; text automation has received the highest priority until most recently. Skill levels and equipment, ranging from word processors to typesetters, are character and not image generators. Also, technical illustraiton is a rare art, and in essence, this new requirement for more graphics can only be met by computer graphic tools. Although these devices readily facilitate the generation and editing of pictorial data, the pagination and large volume manufacturing (text-oriented) processes will have difficulty in generating 'New Look' or 'Job Guide' pages. In short, the problems that this new initiative poses are substantial.

F. Text and Graphic "Commonalities"

Conceptually, it is not difficult to find commonalities for the treatment of text and graphic data structures. The parameters are, on the one side, encoding graphics as is currently done for text, while supplementing the encoded graphic character stream with typographic and presentation attributes. On the other side, it is possible to treat text as line segments, making no distinction between characters and objects, and defining an image as a particular pattern of dots on a page. The current debate in the computer graphics industry between ''rasterized'' and vector data is part of this issue. Data represented by line segments and dot patterns are not in an editable position (as compared to character/vector data). Therefore, during the creation and editing stages of technical information presentation, it is desirable to maintain graphics data in a vector form and alphanumeric data as encoded characters. Once the editorial segment is complete, it is then feasible to raster scan the data and treat it in this manner. In any case, data structures and treatment should be ideally suited toward the application. The problem that still remains is how to manufacture a page through a process that handles data in different formats.

G. Raster Image Processing

A page image can be defined as a pattern or grouping of black dots on white background. The dots can be generated by an electrostatic plotter or a laser sweeping horizontally (or vertically) across the page. The laser is modulated to expose the presence of dots at a fixed resolution. Thus, x number of strokes can create any given image on the page. As an output mechanism, the plotter or laser does not distinguish between graphics and text (although in terms of building the image it has to access and develop the individual characters). The advanced laser or computer-to-plate technology that is currently being refined for newspaper and large printing applications is built around a raster image processor that can accept alphanumeric or vector data, build a mater typographic image, scan, convert, and deliver that image to a plate. Commonality is found at a post-mastering state where the data, be it text, logos, line art, halftones, is projected as a dot pattern for press imprint. What is required is the capability to develop an image from encoded data, scan the image to produce raster line segments, and generate the image at typesetter-quality resolution. The image generator thus strokes the page horizontally and vertically working *from* an image. One typesetting device in particular, the Monotype Lasercomp, can either create an image from a raster digital font library or accept an image, process the image into a series of line segments, and generate the image from a raster scan. The raster image processor can additionally be instructed to prepare an image for various stroking and scanning densities. Both Information International and Autologic are developing raster image processing devices that accept typographically marked up alphanumeric data and output raster scan data at various writing resolutions.

Commonality of data streams at this level comes with a price. It is obviously expensive, even with data compression techniques to store and transmit rasterized data. Encoded character transmission at 9600 baud, as an example, is a magnitude faster than sending a similar page image in a raster scan at the same rate. On the processing side, the data (the page image) is not in an editable format and cannot be recomposed or manipulated. Work has been underway to convert raster scan data into vector for editing on computer graphic terminals, but a product of this type has yet to find its way into a production environment. Data structure intelligence is sacrificed for commonality. This treatment, however, is totally dependent upon the operating environment and, as an example, in a computer-to-plate process, assumptions are that the editorial process has been completed and that the image exists in its totality (paginated, text and graphics logos). It has also become more obvious that as more ''mastering'' (typographic imaging and pagination) occurs on interactive graphic display systems that edit and integrate both text and graphics, the typesetter will function only to replicate the master image at a higher resolution. This ''screen dump'' processing has already occurred on smaller systems. Imagen, a California-based company, is marketing a raster image processing device, the IMP-10, which can be coupled to the Cannon electronic printer. The intent is to accept an image developed on an interactive graphics display terminal, convert the graphic display into raster scan, and then drive an output device from the scanned data to reproduce that image at variable (but fixed) resolutions. XEROX has been working on this technology for some time and currently offers EPIC, a graphics handling software package. EPIC not only generates raster scan data for the electronic printers, but also compresses the data to meet the monotonic processing requirements of the electronic printing systems. This product should be of particular attraction to the technical documentation community, since commitments have been made to use electronic printing systems for the production of technical documentation. EPIC,

In case of conflict between any statement in any article and the standard, the standard governs.

in conjunction with the XEROX 1050 graphics scanner, can scan a rastered image and present the data in a ''Calcomp-type'' format to drive the electronic printers.

Raster image processing, therefore, seems best described as a post-mastering way of treating text and graphic data. Output processing speeds and writing resolutions continue to improve, but problems remain, again, with the storage and communication of line segment data. Once in a rasterized format, the data cannot be easily edited and manipulated with any degree of processing intelligence. Images can be enhanced, cropped, scaled, colored, sized, etc., but not edited. Raster image processing does, however, offer the commonality of data structure that blurs the text/graphics distinction. This can, therefore, offer significant enhancements to the manufacturing of technical information as it is practiced today.

H. Graphic Primitives

The second approach lies on the definition of graphic primitives and graphic font characters. Instead of representing a graphic image as a series of black and white pixels, it is possible to describe the same image as a series of graphic characters in a certain configuration. As alphanumeric characters are labeled and encoded for machine processing, graphic characters are labeled and encoded for machine processing, graphic characters (lines, arcs, curves, polygons, angles, etc.) can also be uniquely expressed in code for transmission and presentation. Graphics standards committees have been working for some time to develop a set of graphic primitives for object representation. The primitives could then be encoded with attributes (size, scale, xy coordinate points, color, etc.) to represent any graphic image from a flow chart to an engineering drawing. More recently, the Air Force, working with the major computer graphic manufacturers and users, has developed methods by which graphic data files can be transported between dissimilar systems for display and editing. The Initial Graphics Exchange Specification (IEGS) is significant to the technical information community because it addresses the transmission of graphic images in both a display image and editable form. Basically, IGES has developed a ''metalanguage'' by which a graphic presentation can be defined. The definition identifies the graphic objects of the image, their scale (sizing, depth), and the corresponding grid coordinates to the objects that are part of that image. What this entails is a post-processor on a graphics system that can convert source specific presentation and attribute code into this metalanguage descriptor file, and a preprocessor on the receiving system that can interpret the metalanguage into that system internal processing language. Document manufacturing is a two-dimensional process (IGES is for two and three) and therefore the extent is to reproduce the image. The encoded graphic data would be treated as text, with markup commands to instruct the typesetter on the positioning and representation of the graphic objects to construct the image. Obviously, this metalanguage description is device independent, and with specific preprocessors, can be used to drive any output device.

An American National Standards Committee (ANSI X3H3) is currently developing a Vertical Device Metafile for graphic data. The objective of this proposed standard is to provide for ''the display and storage communication of graphics information in a device independent manner.'' This includes a methodology for the exchange of picture data between computer graphic installations and devices. The ingredients of the proposed standard include a method of defining, through a metalanguage, the transfer tool, graphic primitives for describing the picture, attributes for varying the appearance of the primitives, and control information for the regeneration of the initial display. The definition of graphic primitives is significant, since these same graphic characters can be plugged into a typesetter for the regeneration of a graphic image. The typesetter could, therefore, accept encoded pictorial data and output the image at the same or higher resolution. The picture image could also be merged with encoded text. It would, therefore, be possible to concatenate two encoded files for the pagination and production of technical documentation pages at a consistent rate of speed.

The First Graphics Kernel System (GKS) embodies much of earlier graphic standardization efforts. The initial goal is to define a core graphics system for displaying the pictorial elements of a graphic. The core graphic system should, therefore, provide for a method to (1) define a graphic descriptively, (2) process the descriptive data and (3) interact with a particular graphic system to view the end object. Once a determination of graphic primitive elements and their allowable attributes is made, it is a short step to encoding the data for communication.

GKS will provide for file portability between different graphic systems. Since a typesetter is, in fact, a graphic system, it is feasible for a typesetting process to accept a descriptive metafile of, as an example, an engine schedmatic, and use its own font library of graphic primitives to create the image? This would guarantee system compatibility between computer graphic systems and output devices, probably speed up processing throughout, and ease the communication problem.

I. Industry Consensus?

Industry consensus is that raster image processing holds the key toward resolving the issue of text and graphic compatibility. The growing interest in higher resolution electronic printing and typesetting systems (the IBM magnetic erosion printer and the Monotype LaserComp II) testifies to this belief. It is also true that almost every conceivable front-end system manufacturer is developing a graphics display system for pagination, where both the text and graphics will be displayed and edited prior to printing. These systems

In case of conflict between any statement in any article and the standard, the standard governs.

will be both interactive editors and formatters. There will still be a requirement, however, to interface to the engineering, quality assurance, and validation graphic data bases. Therefore, the GKS will hopefully facilitate transportability among these devices. The question becomes one of where the initial page image will be ''mastered,'' and then, how it will be reproduced. Is it feasible to interactively manipulate each page of a technical order or a manual? To what degree is batch composition desirable, or even required?

The demands upon the technical information-providing community will ultimately determine the most efficient and cost effective way to handle text and graphics, from data structure to pagination. It is, nonetheless, important to consider all available options to achieve the best results, and this requires careful analysis of hardware/software and configurations, customer requirements, and personnel. System compatibility must be achieved; the questions are how, when, and at what price. GCA will continue its work in this area to keep its membership abreast of new developments and applications.

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

Rules & Regulations Via GenCode™

Joseph P. Johnson

Technology Assessment Officer
Federal Deposit Insurance Corporation

Speech presented at
GCA 1983 Spring Conference
Philadelphia, 1983 May 12

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the GCA copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Graphic Communications Association. To copy otherwise, or to republish, requires a fee and specific permission. © 1983 GCA

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

A. Background

The FDIC loose-leaf reporting service (''the Service'') is a two-volume manual containing the FDIC Act, FDIC Rules and Regulations, statements of policy and general counsel's opinions, and related banking laws, regulations, and other material. The Service is printed and distributed six times a year and almost 22,000 copies are in circulation. The Service is often identified as the Rules & Regulations.

The Service is divided into nine parts, separated by tab cards. Each part has its own table of contents, followed by the text. Pages are numbered sequentially, with breaks between the sections to allow for growth. Decimal or point pages are allowed in a double-digit format, for example, 5750.01.

B. Short History

Originally the text was maintained and printed with a hot metal process. Several years ago we converted to an electronic composition process. At the time of conversion we recognized that the coding of textual data produced on computers and word processing equipment has been a problem for many years. However, we did not anticipate the following problems which occurred after the successful conversion:

(1) The composition system was neither robust or extensible. There was extensive use of cut and paste. The vendor had difficulty in maintaining an up-to-date text data base.

(2) The vendor maintained that the composition codes were proprietary. Their view was that the composition system was on the leading edge of composition technology. Thus, any revelation of the codes would place them at a competitive disadvantage.

Example 1 is a sample of the former contractor's text.

Our problem with this software product was not limited to this particular vendor. The problem was symptomatic of an underlying crisis in software development. The old methodologies and tools used by many software developers could not express or execute an intended design of large or complex projects. Some of the general symptoms of this crisis which were reflected in our situation are the following:

(1) Modifiability—a software change is costly, complex, error prone, and often impossible to implement.

(2) Cost—the costs were perceived as excessive.

(3) Timeliness—the system was delivered with less-than-promised capability.

A decision was made to search for a methodology and software program product for text composition which incorporated the software engineering goals of modifiability, reliability, and understandability. In addition, the methodology and software must apply the following principles:

(1) Abstraction and Information Hiding—the ability to extract or represent essential properties while omitting inessential details. The ability to suppress how a process is implemented, and to focus on higher abstractions.

(2) Modularity—the ability to apply a top-down methodology where higher-level modules specify *what* action is to be taken while the lower-level modules define *how* the action is to be carried out.

GenCode™ was selected as the methodology because it promoted good software engineering practices. In particular it supported the following goals:

(1) A generalized approach definable in practical terms of our needs, yet flexible enough to permit future expansion.

(2) Provide for a generalized data exchange mechanism applicable to text data files. The GenCode™ concept permits the transfer of text data sets among different software systems while preserving structural and content integrity.

(3) Facilitate the processing of textual data sets by allowing the production of identical documents in diverse information products and typographical formats.

(4) Support author preparation of the Rules and Regulations manuscript.

(5) Facilitate document assembly from various subdocuments.

The GenCode™ approach permitted us to identify text elements and hierarchies without reference to a specific processing system or output device.

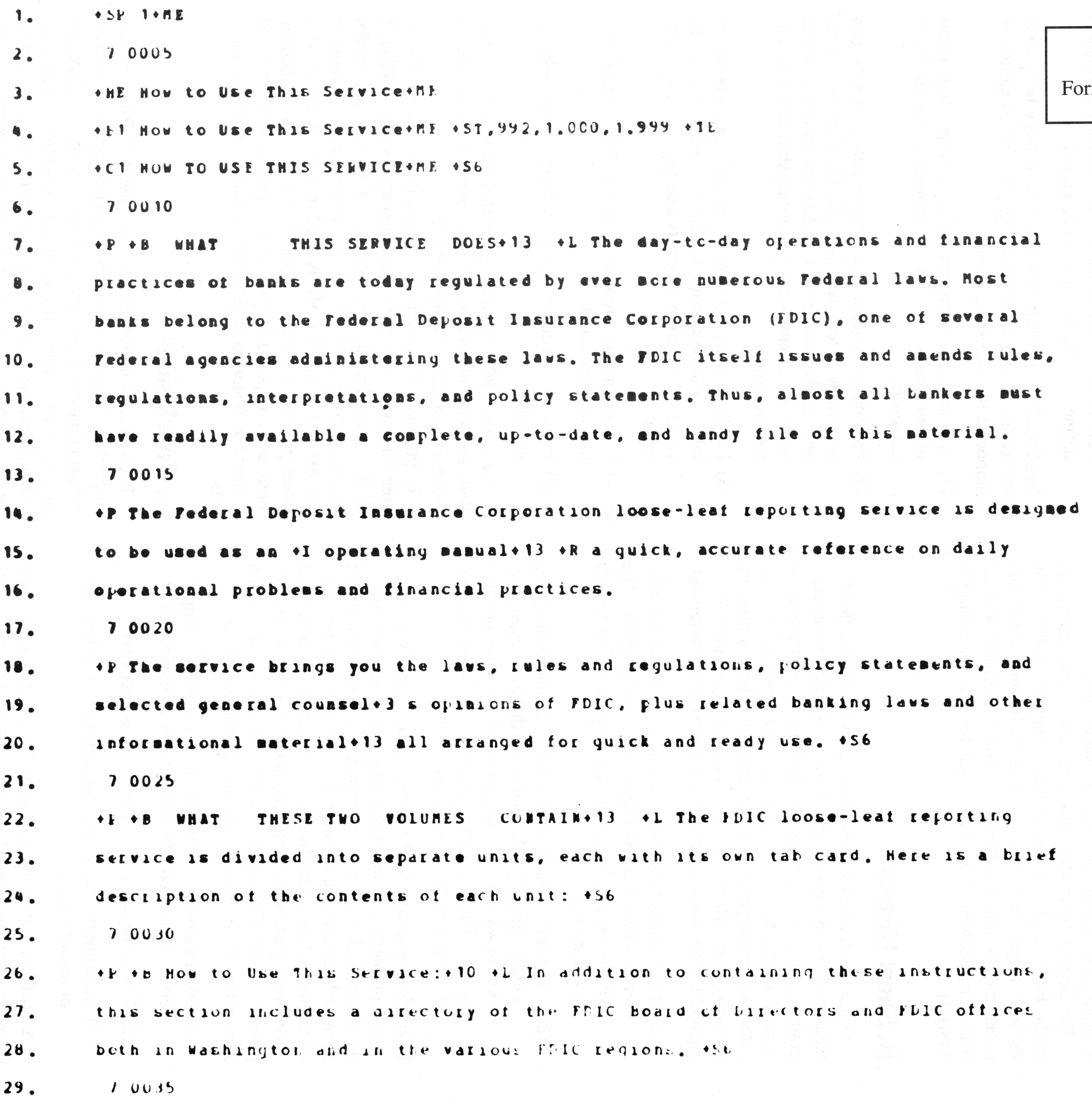
In practice, the electronic composition system, CS IV from Informatics General Corporation and Waterloo SCRIPT GML were selected. The data sets of the original vendor were converted to CS IV. CS IV is now used to compose the text and SCRIPT GML is used to monitor the composition process. Examples 2 and 3 contain CS IV generic input and composed output. In examples 4 and 5 the text is coded in Waterloo SCRIPT GML and printed on a Xerox 9700.

C. Conclusions

There was an unplanned cost in converting from vendor-proprietary composition codes to generic codes. However, our experience is that current and future savings will exceed the cost of conversion to generic text. Current savings were realized because there is now competitive procurement of the loose-leaf service. We believe future saving will be realized. The text can be made available on other print presentation devices without a massive and expensive recoding effort.

In case of conflict between any statement in any article and the standard, the standard governs.

EXAMPLE 1.
Former Contractor's Text



```
1.     ♦SP 1♦ME
2.      7 0005
3.     ♦ME How to Use This Service♦ME
4.     ♦E1 How to Use This Service♦ME ♦ST,992,1.000,1.999 ♦1E
5.     ♦C1 HOW TO USE THIS SERVICE♦ME ♦S6
6.      7 0010
7.     ♦P ♦B  WHAT      THIS SERVICE  DOES♦13  ♦L The day-tc-day operations and financial
8.     practices of banks are today regulated by ever more numerous Federal laws. Most
9.     banks belong to the Federal Deposit Insurance Corporation (FDIC), one of several
10.    Federal agencies administering these laws. The FDIC itself issues and amends rules,
11.    regulations, interpretations, and policy statements. Thus, almost all bankers must
12.    have readily available a complete, up-to-date, and handy file of this material.
13.     7 0015
14.    ♦P The Federal Deposit Insurance Corporation loose-leaf reporting service is designed
15.    to be used as an ♦I operating manual♦13 ♦R a quick, accurate reference on daily
16.    operational problems and financial practices.
17.     7 0020
18.    ♦P The service brings you the laws, rules and regulations, policy statements, and
19.    selected general counsel♦3 s opinions of FDIC, plus related banking laws and other
20.    informational material♦13 all arranged for quick and ready use. ♦S6
21.     7 0025
22.    ♦P ♦B  WHAT   THESE TWO  VOLUMES   CONTAIN♦13  ♦L The FDIC loose-leaf reporting
23.    service is divided into separate units, each with its own tab card. Here is a brief
24.    description of the contents of each unit: ♦S6
25.     7 0030
26.    ♦P ♦B How to Use This Service:♦10 ♦L In addition to containing these instructions,
27.    this section includes a directory of the FDIC board of Directors and FDIC offices
28.    both in Washington and in the various FDIC regions. ♦S6
29.     7 0035
```

Example 2.
CS IV
Generic Input

```
1.    :mp1
2.    *MK0
3.    :hd,2; 12-31-82
4.    :hd,3;How to Use This Service
5.    :hd,4;How to Use This Service
6.    :ao;
7.    *H1 HOW TO USE THIS SERVICE
8.    *BK1
9.    *P3 WHAT THIS SERVICE DOES---*ro The day-to-day operations and financial
10.   practices of banks are today regulated by ever more numerous Federal laws. Most
11.   banks belong to the Federal Deposit Insurance Corporation (FDIC), one of several
12.   Federal agencies administering these laws. The FDIC itself issues and amends rules,
13.   regulations, interpretations, and policy statements. Thus, almost all bankers must
14.   have readily available a complete, up-to-date, and handy file of this material.
15.   *P1 The Federal Deposit Insurance Corporation loose-leaf reporting service is designed
```

Example 3.
CS IV
Photocomposed Output

12-31-82 1

HOW TO USE THIS SERVICE

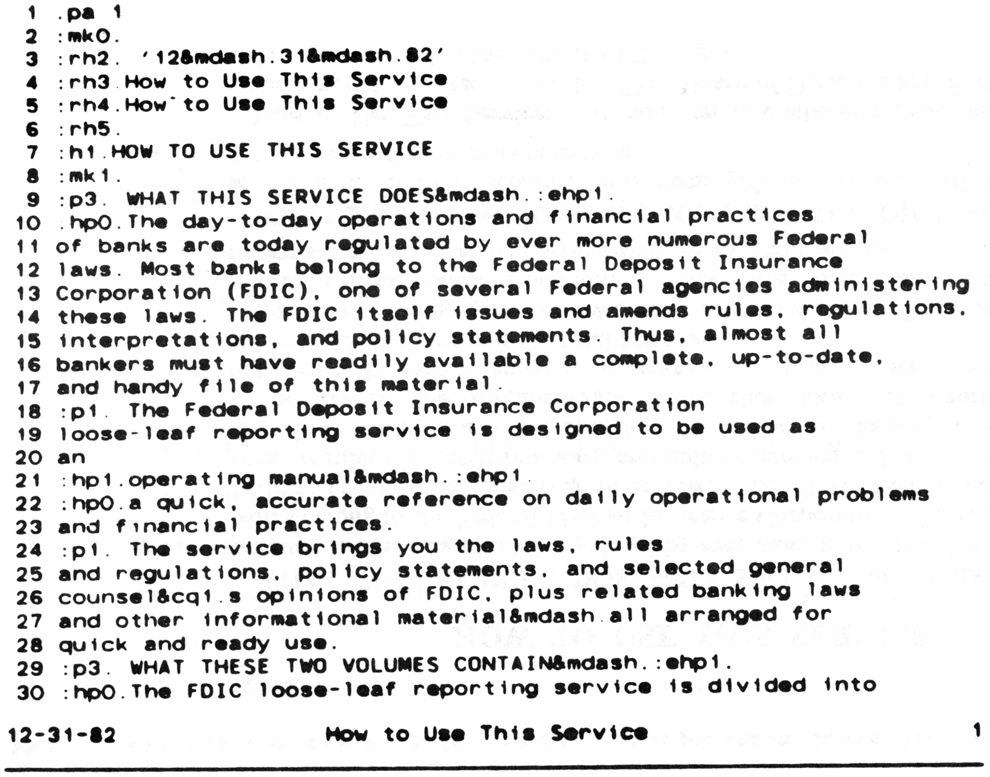
WHAT THIS SERVICE DOES—The day-to-day operations and financial practices of banks are today regulated by ever more numerous Federal laws. Most banks belong to the Federal Deposit Insurance Corporation (FDIC), one of several Federal agencies administering these laws. The FDIC itself issues and amends rules, regulations, interpretations, and policy statements. Thus, almost all bankers must have readily available a complete, up-to-date, and handy file of this material.

The Federal Deposit Insurance Corporation loose-leaf reporting service is designed to be used as an *operating manual*—a quick, accurate reference on daily operational problems and financial practices.

The service brings you the laws, rules and regulations, policy statements, and selected general counsel's opinions of FDIC, plus related banking laws and other informational material—all arranged for quick and ready use.

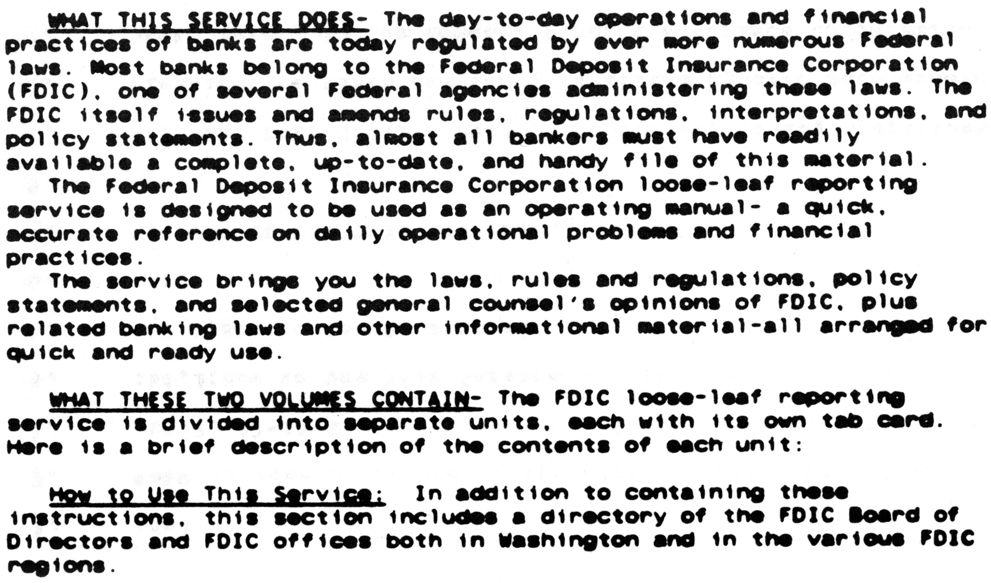
WHAT THESE TWO VOLUMES CONTAIN—The FDIC loose-leaf reporting service is divided into separate units, each with its own tab card. Here is a brief description of the contents of each unit:

How to Use This Service: In addition to containing these instructions, this section includes a directory of the FDIC Board of Directors and FDIC offices both in Washington and in the various FDIC regions.



```
.pa 1
:mkO.
:rh2. '12&mdash.31&mdash.82'
:rh3.How to Use This Service
:rh4.How to Use This Service
:rh5.
:h1.HOW TO USE THIS SERVICE
:mk1.
:p3. WHAT THIS SERVICE DOES&mdash.:ehp1.
.hpO.The day-to-day operations and financial practices
of banks are today regulated by ever more numerous Federal
laws. Most banks belong to the Federal Deposit Insurance
Corporation (FDIC), one of several Federal agencies administering
these laws. The FDIC itself issues and amends rules, regulations,
interpretations, and policy statements. Thus, almost all
bankers must have readily available a complete, up-to-date,
and handy file of this material.
:p1. The Federal Deposit Insurance Corporation
loose-leaf reporting service is designed to be used as
an
:hp1.operating manual&mdash.:ehp1.
:hpO.a quick, accurate reference on daily operational problems
and financial practices.
:p1. The service brings you the laws, rules
and regulations, policy statements, and selected general
counsel&cq1.s opinions of FDIC, plus related banking laws
and other informational material&mdash.all arranged for
quick and ready use.
:p3. WHAT THESE TWO VOLUMES CONTAIN&mdash.:ehp1.
:hpO.The FDIC loose-leaf reporting service is divided into

12-31-82              How to Use This Service              1
```

EXAMPLE 4.
Text Coded In
Waterloo SCRIPT GML



```
   WHAT THIS SERVICE DOES- The day-to-day operations and financial
practices of banks are today regulated by ever more numerous Federal
laws. Most banks belong to the Federal Deposit Insurance Corporation
(FDIC), one of several Federal agencies administering these laws. The
FDIC itself issues and amends rules, regulations, interpretations, and
policy statements. Thus, almost all bankers must have readily
available a complete, up-to-date, and handy file of this material.
   The Federal Deposit Insurance Corporation loose-leaf reporting
service is designed to be used as an operating manual- a quick,
accurate reference on daily operational problems and financial
practices.
   The service brings you the laws, rules and regulations, policy
statements, and selected general counsel's opinions of FDIC, plus
related banking laws and other informational material-all arranged for
quick and ready use.

   WHAT THESE TWO VOLUMES CONTAIN- The FDIC loose-leaf reporting
service is divided into separate units, each with its own tab card.
Here is a brief description of the contents of each unit:

   How to Use This Service:  In addition to containing these
instructions, this section includes a directory of the FDIC Board of
Directors and FDIC offices both in Washington and in the various FDIC
regions.
```

EXAMPLE 5.
Text Output On
Xerox 9700

GenCode™ Pre-Processor Requirements

Jordan J. Glogau

Executive Vice President
Medianet, Inc.

Material originally presented at the
GCA 1983 Spring Conference
Philadelphia, 1983 May 12

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the GCA copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Graphic Communications Association. To copy otherwise, or to republish, requires a fee and specific permission. © 1983 GCA

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

TABLE OF CONTENTS

A. Areas of Concern 231
1. Physical Form 231
2. Coding and File Management Schema 231
3. Editorial Structure and Content Element Identification 231
4. Pluses and Minuses Along the Path 232
B. Word Processor Advances 232
C. Generic Preprocessor Requirements 232
1. Operating System Requirements 233
2. Preprocessor System Requirements 233
3. Composition System Requirements 233
D. Preprocessor Requirements in Detail 233
1. Program Nature 233
2. Pattern Recognition 234
3. Pattern Definition 234
4. Content Patterns—Processing By Line 234
5. Interactivity 234
6. Structure 234
E. Preprocessor Program Modules 234
1. Input Modules 234
a. Input Module 1 234
b. Input Modules 2 and 3 234
c. Input Module 4 235
2. GenCode™ Modules 235
3. GenCode™ Special Modules 235
F. Composition System Module 236
G. Coding Examples 236
H. Summary 236

In case of conflict between any statement in any article and the standard, the standard governs.

TABLE OF CONTENTS (*Continued*)

LIST OF FIGURES

1. Preprocessor Program Modules 237
2. Input Program Modules 237
3. GenCode™ Program Modules 238
4. Composition Program Modules 238

LIST OF EXAMPLES

1. Text File with No Coding 239
2. Text File with Word-Processor Coding 239
3. Pattern Recognition 240
4. Text File with GenCode™ Coding 240
5. Typographic Specifications 241
6. Text File Set to Typographic Specifications 242
7. Text File with Typical Publisher's Coding 242

In case of conflict between any statement in any article and the standard, the standard governs.

With the explosion of information technologies today we have created an electronic Tower of Babel. We are generating text on every conceivable type of word processor, personal computer, and main frame device, and we have no reasonable way to talk to one another or exchange information from one system to another. This situation is going to get worse because there are increasing numbers of vendors using different ''standards'' and approaches to putting information out electronically.

A. Areas of Concern

If one analyzes the requirements of the *interchange* process, three principal areas of concern emerge:

(1) **Physical Form**
The electrical and mechanical dimensions of the source machine and the distination machine, and any interface conversion black box or telecommunications channel which may exist between the two machines, must all be compatible.

(2) **Coding and File Management Schema**
The meaning of particular codes must be mutually understood—e.g., is an EBCDIC ''A'' the same as an ASCII ''A''?

(3) **Editorial Structure and Content Element Identification**
One must be able to identify that a certain character string group is a heading while another is a paragraph text. Furthermore, the identity of a chapter title as contrasted to a heading must be established. Most importantly, one must be able to identify these character strings in a manner that is completely machine-independent. The strings must be clearly and immediately recognizable within either the source machine or the destination machine without special or additional processing. Consequently one must identify the elements of content that establish the editorial hierarchy, which defines the very structure of written communication.

1. Physical Form

A new home computer system from Coleco Industries includes a printer, a keyboard, a word-processing software package, and a local removable storage device, and costs under $600.00. The storage device is not a floppy disk; it is an endless-loop cassette that will be able to record information for later editing and printing. Significantly, companies like Atari and Coleco are expecting to sell over 10 million devices of this type this year alone, with the result that more and more people will be using them to generate information that they will expect to output to a typesetter. Coleco's endless-loop cassette represents yet another medium that printers and typesetters will need to accommodate on their systems.

2. Coding and File Management Schema

However, even if the typesetting equipment can physically accept the electronic storage medium, or if you can telecommunicate the file from one device to another, you are really addressing only the first part of the problem. The second part concerns the translation of the codes. Every device has its own unique set of codes, character set, file management, file storage capabilities, and methods for recording and retrieving information.

The problem that the destination system has with accepting information from a source system means that the information has to be converted by the destination system into the format that it can understand. A good analogy to this problem of *transferring* and *translating* information is to think of one person who speaks English and wants to write a letter to another person who speaks Spanish. The first step is to write the letter in English and mail it to the person who speaks Spanish. This is the *transfer*. The second step is for the person who wants to be able to receive it, and who speaks only Spanish, to give it to an interpreter who *translates* it into Spanish. The second person is now able to understand the message that the first person was trying to communicate. The conversion system is like the *translator* or the *interpreter* that does this particular task.

3. Editorial Structure and Content Element Identification

One must be able to identify that a certain character string group is a heading while another is a paragraph text. Furthermore, the identity of a chapter title as contrasted to a heading must be established. Most importantly, one must be able to identify these character strings in a manner that is completely machine-independent. The strings must be clearly and immediately recognizable within either the source machine or the destination machine without special or additional processing. Consequently one must identify the elements of content that establish the editorial hierarchy, which defines the very structure of written communication.

Futurists in GCA began talking about generic identification methods ten or more years ago. The various committee discussions, conferences, and seminars since that time have led to the development of GenCode™.

The GenCode™ Standard establishes the ability to:

(1) create information on one system and, with a minimal amount of difficulty,

(2) send it to a second system, and

(3) receive it at that second system without extensive problems of interpreting it on the second system.

An important consideration arising from the development of GenCode™ involves whether the practical difficulties caused by the lack of standardized interchange between word

In case of conflict between any statement in any article and the standard, the standard governs.

processing systems—which has given rise to the black box conversion segment of the industry—will stand in the way of GenCode™ implementation.

4. Pluses and Minuses Along the Path

The good and less good aspects of adopting GenCode™ remind me of a story about a philosophy student who, after six months of taking his first course, walked up to his philosophy professor and said, ''Professor, I'm completely confused. It seems that everything that you've said about philosophy is a paradox.'' ''Well,'' the professor said, scratching his chin, ''Yes and no.'' On the ''Yes'' side, the proliferation of different devices and the present lack of standards have promoted a learning curve within different segments of the information processing industry about the importance of equipment communication. The first people who woke up to the fact that there was a need for a *standard of exchange* were organization like the Internal Revenue Service and Standard & Poor's, both of which had large data bases containing existing information that could be used in other mediums, such as on-line, Braille, and microfilm. Large printers, typesetters, and publishers, all of whom would benefit from the standard, subsequently started to take an interest in the mid-70's in getting GenCode™ off the ground. Even medium-sized typesetting houses started to see the need to have this same freedom of exchanging information from one system to another.

Currently smaller typesetting firms are interested in having the ability to exchange generic files, and although consumers with Atari or Commodore computers are not yet thinking about this as an issue, they too will soon want to be able to get all kinds of information in and out of their system without a lot of problems. When that occurs, the size of the data base that people will want to exchange shrinks to a personal letter or resumé, and it is at that time that GenCode™ will be accepted.

On the ''No'' side, the major obstacle to GenCode™, is that it is structured so that most people are not going to want to be bothered with the burden of the coding of their documents. Most people will see coding as a disadvantage because most word processors don't rely on codes to format the printed output; any layout changes appear immediately on the screen. When people do a numbered list or a tabular job they don't want to have to do some of the coding that GenCode™ requires. The politics of getting people to code is an important factor in the acceptance of any coding, GenCode™ or otherwise.

This reaction to coding is important because publishers, the federal government, advertising agencies, and almost every other segment involving people who create text, will have a different reaction to the necessity of coding jobs. Acceptance of generic files tends to occur, however after people discover what they have to go through when they want to transfer one file into a second format that isn't standardized.

B. Word Processor Advances

Acceptance of GenCode™ will also be promoted by the fact that word processors are becoming more and more sophisticated in terms of their control functions. A review of how tabular work has been done on devices between 1976 to 1983 will illustrate this point. The first popular word processor, the Vydec, was a good computer for its time, except that if you ever tried to get any information from the Vydec other than straight text you had many problems. There was no easy way to capture tabular from the Vydec word processor. There was absolutely no coding between columns that would indicate that you wanted to go from the stub to the next column and to the next column. You had space bands, but if you had regular text mixed in with columns, you could not tell the difference between a regular space band that would occur in text and a space band that would occur between columns.

The first big advance came with the Wang word processor, which allowed one to place tab codes between columns so that information would align aesthetically in a fashion similar to the result you get when you set a margin and tab stops on a typewriter. These codes, which were binary codes, were imbedded between the columns within the text stream. A newer device that comes to mind is called the Syntrex. This computer allows you to define different kinds of tab actions, including right aligning, left aligning, and aligning on a decimal. These tab codes, as with the Wang, are imbedded within the file stream.

C. Generic Preprocessor Requirements

The preprocessor that is needed for a GenCode™ file is going to be one that doesn't just address GenCode™ in a purist fashion. It will, however, be ''generic'' enough to be able to address all the requirements of different input devices and all the requirements of different output devices.

The requirements of a generic preprocessor can be divided into three major areas:

(1) the operating system,

(2) the preprocessor, and

(3) the composition system.

1. Operating System Requirements

If the operating system is multi-pathed, and if the preprocessor is integrated with a composition system and running in a multi-processing environment, then the system could be doing composition and preprocessing at the same time. In this situation the operating system should be flexible in terms of its input and output. It should be able to accept magnetic tapes and telecommunications (both asynchronous and bisynchronous) and be able to interface, without many problems, to the various black boxes that make media and telecommunications conversions, such as those manufactured by Shaffstall and Applied Data Communications. The device handlers (that is, the particular software that will handle the input/output (I/O) routines for the I/O, magnetic tape, telecommunications, and the black boxes) should be able to work with different character sets, such as ASCII and EBCDIC. The software should be able to do simple character replacements and delete the most obvious garbage, such as null characters and unwanted file headers. Finally, the operating system should allow you to have a ''job control language,'' which allows one to scan five or six small files into the system in raw mode (which is information before it has gone through the preprocessor) and send them all through the processor, composition, and pagination steps unless an error is flagged in one of these processes. All of these steps proceed without requiring the user to execute different commands for each processing step.

2. Preprocessor System Requirements

In the context of considering all preprocessor requirements—i.e., those associated with physical form, coding schema, and file management schema, as well as the editorial structure and content element identification normally associated with GenCode™, we are establishing a broad overall system requirement. This overall system must, therefore, deal with the problems of transfer and translation as well.

This approach, therefore, treats the GenCode™ requirements, in effect, as a subset of the over-all preprocessing system.

3. Composition System Requirements

As an absolute minimum the composition system should support formats, which are a short combinations of alphanumerics that correlate with larger strings of specific typesetting commands. People have different names for formats, such as locator code, style, and macros. Formats are like subroutines in programming languages; instead of having to repeat certain functions that are done again and again, a subroutine is called. Formats are important because they can be used on a one-to-one basis with the Generic Identifiers (**GI's**) that are part of the GenCode™ Standard. **GI's** will divide a document into a series of elements of like and unlike kind and assign to each one of these elements different graphic parameters, such as font, leading, and point size.

The composition system should also offer note suppression. The system should be able to extract a generic file out of a composition file, eliminating the need for dual files, e.g., the generic file and the composed file as two separate data bases. With note suppression, generic information preserved in the file won't be typeset, but it *will* be resident for later uses. It would be even better if note suppression could be multi-leveled. For example, one level would be appropriate for composition coding, another level might be used to keep information needed for an interactive data base, and a third level could serve for a fixed data base for something like Teletext.

The third requirement of the composition system is the ability to flag errors when something has gone wrong in the preprocessor or the composition stages.

Finally, if the system supports pagination, it should be able to record *back* into the generic file where the page breaks are. This can be very important in certain kinds of data base work. For instance, if you have a printed looseleaf manual, you may also want to have it on-line. It is often a requirement that the on-line data base have references to the number of the page on which the information is printed.

D. Preprocessor Requirements in Detail

1. Program Nature

The preprocessor should be a compiler or interpreter, which means that instead of programming in a low-level language (which is much more time consuming and is prone to error) applications programs under the preprocessor should be written in a higher-level language. High-level language compilers are easy to write, have good diagnostics, and have a full repertoire of error messages. The compiler should also be a structured program. The concept behind structured programming is that if you write 1,000 lines of computer code and one line is written incorrectly, you must search all 1,000 lines in order to figure out where the error is. If it's coded the way most programs are, you are going to be looking all over the place trying to figure out where the mistake is. The structured programming suggests that instead of writing 1,000 lines into a single program the programmer should divide the program into ten smaller sub-programs that accomplish the same task and are related to one another. If there is an error, it will be easier to find because it will be isolated within one of those 100-line sub-programs. A second major concept behind structured programming involves structured design. Structured design refers to the relationship between the sub-program and is similar to setting up a good organizational chart for a company.

2. Pattern Recognition

Another major requirement of the generic preprocessor is pattern recognition. Pattern recognition is best described using a typical typesetting job, such as a numbered list. Most people create a numbered list on a word processor in the following sequence: they skip a space, hit the number, hit a period, and then hit the ''indent tab code'' which most word processors have today to allow the creation of a temporary left-hand margin. (Even if the word wraps, it will wrap on that temporary left-hand margin.) In other words, you do not need to space over for the runover line. Pattern recognition operates in a similar manner by identifying a space followed by any number followed by any period and the generic word processing indent code. If it finds this sequence it replaces that first space with a code for figure space (emphasis by form), then the number, then the period, then the typesetting command or the generic command if the operator felt that was necessary for the indent command on the typesetting system.[1]

3. Pattern Definition

In addition to pattern recognition, the preprocessor needs pattern definition. Certain standard patterns, such as a numbered list, could be found over and over again in different types of documents. A label could be assigned to the pattern for later use.

4. Content Patterns—Processing by Line

The system should also have the ability to process a record of any size, whether it be a line, a paragraph, or a page. Most information is transmitted from a typical black box or magnetic tape unit as a logical record of the physical device. Although this record might be a fixed number based on the recording characteristics of the device that you are receiving the information from, it is not how you want to process that information intellectually. Instead, you want to transfer that information into your system raw and then process it based on the *intellectual* patterns of the file. In most cases, this means processing a file by line or by paragraph, a decision that will depend on the word processor you have. Some word processors, like Wang's store no *soft returns*. If you look at an entire paragraph and it's five lines on the editing screen internally, the first four lines have no special code at their end because Wang word-processing software automatically accomplishes word wrap. The only noticeable special code occurs on the last line where there is a hard return. Other devices will supply you with both hard and soft return, but the preprocessor needs to, and should be able to, process based on *an intellectual entity* as opposed to *a physically recorded entity* such as a record or track slice from a floppy disk.

Along with being able to process by lines, you should be able to recall previous lines and process them based on new information.

5. Interactivity

Finally, the preprocessor should be interactive. It should allow you, through the preprocessing, to query patterns that are incorrect and adjust them as they are found. If it can only process in a batch mode, it should be able to put an error code out, which might be something you could see on your composition system's editing terminal.

6. Structure

The structure of the generic preprocessor would then be set up with program modules as shown in Figure 1. The executive would be the controlling file that would bring the files in and send them back out once they've been processed. Its three main modules would be the input module, the GenCode™ module, which would actually deal with the generic coding structure, and the output or composition module. The input module would deal with the specifics of the particular device that you're bringing information from. The GenCode™ preprocessor module would deal with the hierarchial structure of the GenCode™ generic identifiers. Since it is a generic preprocessor, the GenCode™ preprocessor could be substituted with another publisher's coding scheme. Finally, the composition module would be set up so that the GenCode™ module would actually turn out generic typesetting files that were very similar to generalized typesetting files, whereas the composition module would turn out the specific codes for a particular kind of typesetting system.

E. Preprocessor Program Modules

1. Input Modules

The input modules deal with the structure of the I/O device codes from which the text file has been received. Each word processor and computer has its own code set and file structure that must be considered. The input module has four working submodules (see Figure 2). These submodules generate a generic text file that is transmitted to the GenCode™ module for further processing.

a. Input Module 1

The first input module cleans up the garbage that is passed by each word processing system along with the text. An example of this garbage is margin controls; most word processors have a string of codes imbedded in the file to tell the word-processing system where the left-hand and right-hand margins and the tab stops are. These codes are generally useless to a GenCode™ or typesetting file. The input module, therefore, sifts out the unneeded coding that will clutter a generic file.

b. Input Modules 2 and 3

The next two modules deal with standard problems that always arise in conversion. One module recognizes quotations because word processors do not distinguish between opening and closing quotation marks, whereas typesetting systems do. The third module deals with lower case ells (l) that should be numeric ones (1). Most older typewriters never had a key for numeric 1's, so typists used the lowercase ells (l) instead. Of course, when the manuscript was rekeyed in the traditional manner, this was manually corrected. But if the manuscript is electronically transferred, the wrong character is typeset (l—ell instead of 1—one). Both of these problems can be corrected with pattern recognition.

c. Input Module 4

The last input module deals with the control codes of the particular word processor that *do* have meaning in the generic-coding/typesetting process. Most of these are the implicit codes available to the word

In case of conflict between any statement in any article and the standard, the standard governs.

processing operator that in some manner change the graphics output on the screen or printer of that system. A partial list of these codes includes:

(1) Underline (double, single)

(2) Bold

(3) Superior/Inferior

(4) Tab
  (a) Tables
  (b) Indent
  (c) Decimal Alignment

(5) Bold (Double printing)

(6) Footnotes

(7) Notes

(8) Center

These codes should and can be used in place of explicit coding to indicate changes in the state of the graphic or generic appearance of a text file. For example, underlines in traditional publisher markup have always indicated to the typesetter that this text should be italicized. This module would adhere to this tradition, except that it would change the underline code to a generic "2nd-level emphasis" code. The reason this last module is needed is that most word processors use different coding sequences for the same function. For example, one word processor may indicate words to be underlined with a delimiter code before and after while another will have a backspace and underline character after *every* character to be underlined. These must be changed to one generic code sequence. Some of this work can be reduced if the word processing conversion device *itself* sends out one generic word processing file format instead of the individual codes of each device. At present, the only black box that does this completely is the Antares.

2. GenCode™ Modules

The GenCode™ modules (Figure 3) take the generic input file created by the input module and generate a generic typesetting file. In GenCode™ document creation, a document is divided into zones such as front, body, and back matter. Each of these zones is subsequently divided into elements such as head, tables, and basal text. This division is analogous to the creation of a very detailed table of contents of the document. The first working GenCode™ module is a table that is a program description of all of the kinds of generic elements in this document. A complete table can be loaded with the specific generic codes in a batch processing environment, or the document can be examined interactively and loaded as the document is processed. The table would direct the file to be processed by the submodules in the appropriate manner. The first working GenCode™ submodule is the implicit coding module. This may seem repetitive of the input module, but the input implicit submodule, in contrast to the GenCode™ implicit submodule, is not concerned with the zone, or state, that the file is in. For example, the input implicit submodule will change underlines to one generic code, but that one underline code could stand for different things within different zones. Within running text it would be a "1st-level emphasis" code (e.g., a font change to italic), while within tabular it *would* be an *underline* to indicate the start of the body of a table after the column heads. Furthermore, an underline of a complete "paragraph" could indicate a head instead of text. In the last use of the implicit underline code, the zone is actually changed within the text stream. In this case, the implicit codes change the table pointer, which would, in turn, direct the lines in this newly defined zone to be processed in the appropriate code, once the zone has been determined by explicit or implicit coding. The table will direct the line to be processed by the generic code classification module. The main classes that would be part of this module would be:

(1) Straight matter

(2) Lists
  (a) Numbered
  (b) Unnumbered
  (c) Bulleted

(3) Tabular
  (a) Regular
  (b) Boxed

(4) Equations*
  *Not previously part of GenCode™

For example, if the GenCode™ called for a numbered list, the GenCode™ text class module would start a counter and add this number to the beginning of each line along with a period and an en space. As new codes were processed, the table pointer would change and different lines would be processed by the different sections of the GenCode™ text classification module.

3. GenCode™ Special Modules

Finally, there are some things that just don't fit into any general category. These zones would be handled by the GenCode™ special module. The table would also include instructions that certain code/zone combinations would have special treatment and they should be sent to this custom module. This should be the place where most of the job-dependent software processing should take place.

The GenCode™ module set could be designed to output a GenCode™ file *or* a generic typesetting file. A generic

In case of conflict between any statement in any article and the standard, the standard governs.

typesetting file is possible because modern composition systems have formats and format merge commands.

F. Composition System Module

The composition modules, Figure 4, would take the output of the GenCode™ module and turn out a specific composition system's coding. The composition module would take the generic typesetting file and replace mnemonic code, such as "**⟨em⟩**," through insertion of the binary code for an em space *for that particular system*. It might seem like this step could take place in the GenCode™ module, but many shops have more than one typesetting system or change technology often enough to merit this as a third step. Also, many shops that start to use the GenCode™ approach find it has merit beyond any particular system they may be using. Therefore, the third module is needed to maintain system independence.

G. Coding Examples

How a generic preprocessor would work can be seen by examining how a preprocessor might work on a small file. This small file will be shown uncoded, coded for word processing, GenCoded™, and coded with a typical publisher's codes. It will also be illustrated in set form, illustrating the use of the typographic parameter table.

Example 1 shows the uncoded file as it would appear on a word processor printout. This would be considered a "what you see is what you get" (WYSIWYG) file. At first glance this file would appear to be uncoded until we superimpose the "hidden" word processing control codes, as can be seen in Example 2. With these codes shown as they "appear" in the serial stream of the text, we can discern the patterns shown in Example 3.

For a file with moderate style changes, implicit coding can be used to generate a GenCoding™ without any "coding" whatsoever. Example 4 displays the same file with the correct GenCodes™. The author wants neither to oversell nor to undersell implicit coding as a substitute for explicit coding. In the real world usually *both are used successfully in combination*.

Each element in Example 3 also illustrates the type of description (in plain English) that would be used in the generic processor to modify a generic file into a typesetting file. The third column needed to commit the file to typography is symbolized by the representative typographic parameter abbreviations shown. Example 5 depicts a representative table of typographic specifications. The text file, set to these specifications, is illustrated in Example 6.

Finally, Example 7 shows a typical publisher's coding. As can be seen, there is little difference between this and the GenCode™ file.

H. Summary

In summary, *generic coding is a pragmatic and economical approach to file/system exchange*. The proper use of GI's is the strongest way of coding for word processing conversion. It is actually *better* than the coding schemes that most publishers use.

The only salvation to this electronic Tower of Babel, therefore, to use an agreed-upon "black box." GenCode™ is the "box" we need because it is flexible enough to be designed to meet *each* End User's requirement.

The generic preprocessor approach has been used by the author on at least five different systems using three different programming languages. Investing in a generic preprocessor instead of doing custom-conversion software for each new job has saved many hours of programming.

Endnote

1. Unfortunately, none of the major programming languages that exist today, including COBOL, FORTRAN, and PASCAL, is built for pattern recognition. The only recognized high-level language that incorporates pattern recognition specifically for the manipulation of text, as opposed to numbers, is a language called SNOBOL, which was written in the '60s by Bell Laboratories. There are not too many people who know about it, and I don't think there are more than 600 sites in the United States that even have the compiler. Datalogics and Rayport, however, offer typesetting systems that do not support the SNOBOL programming language. SNOBOL is a very flexible language; it is supported on the DEC/PDP-11 family of computers, on the DEC VAX, and on the IBM 360/370 family. The problem with SNOBOL, however, is that, even though it is inexpensive to purchase, it is difficult finding a programmer who knows the language.

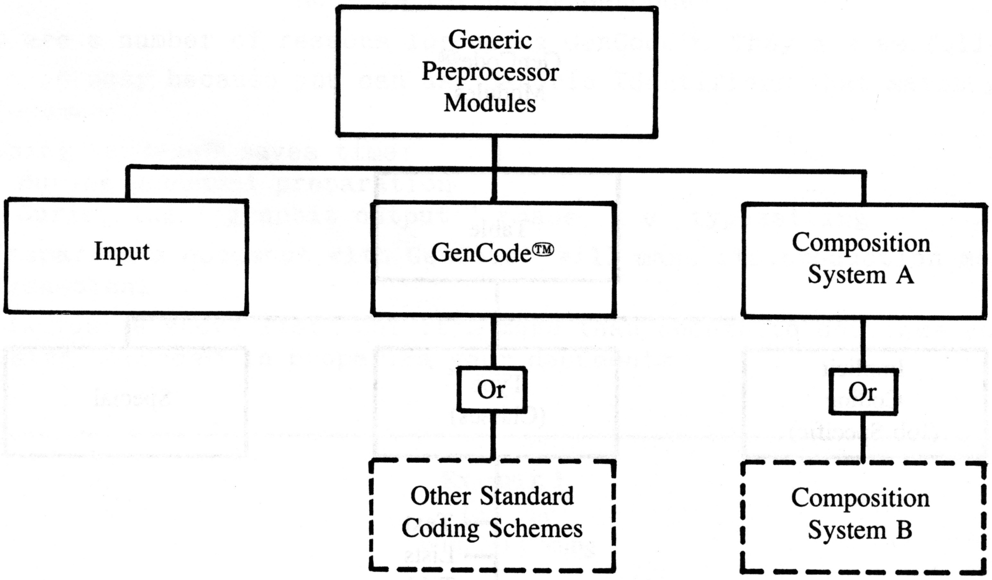

Generic Preprocessor Modules
Input
GenCode™
Composition System A
Or
Or
Other Standard Coding Schemes
Composition System B

FIGURE 1.
Preprocessor Program Modules

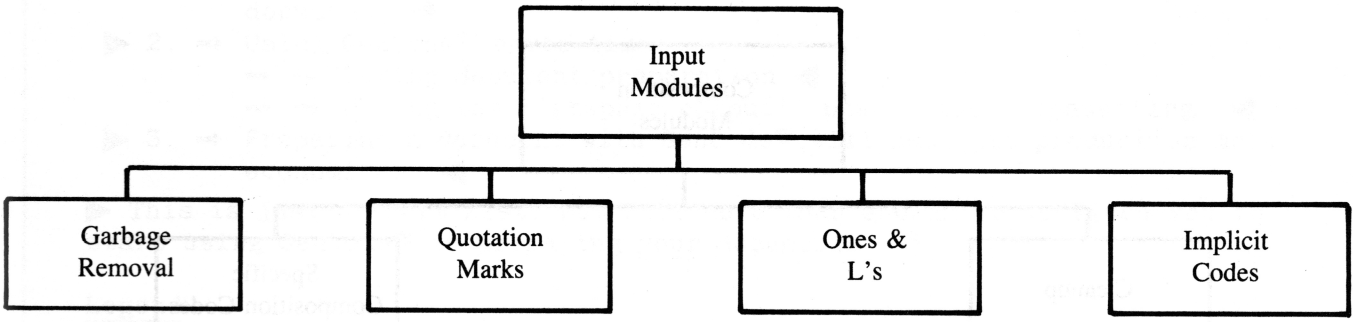

Input Modules
Garbage Removal
Quotation Marks
Ones & L's
Implicit Codes

FIGURE 2.
Input Program Modules

In case of conflict between any statement in any article and the standard, the standard governs.

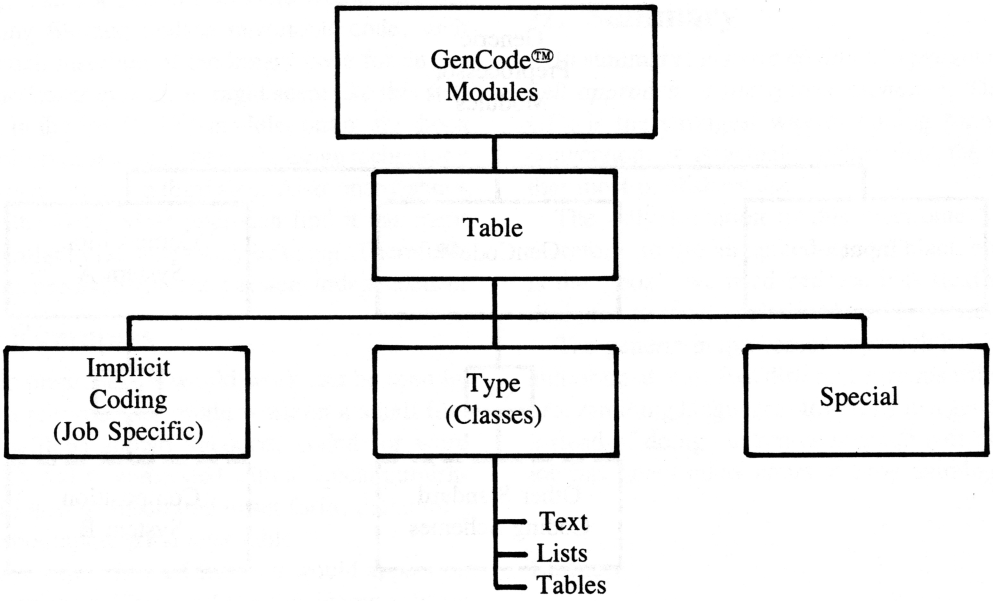

GenCode™ Modules
Table
Implicit Coding (Job Specific)
Type (Classes)
Special
Text
Lists
Tables

FIGURE 3.
GenCode™ Program Modules

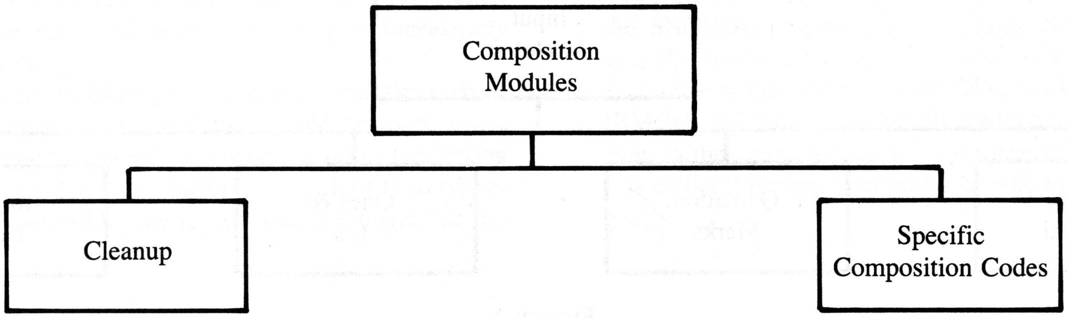

Composition Modules
Cleanup
Specific Composition Codes

FIGURE 4.
Composition Program Modules

In case of conflict between any statement in any article and the standard, the standard governs.

```
                      Reasons for Using GenCode™
   There are a number of reasons for using GenCode™. They are as follows:
    1. It is easy because you can use Generic Identifiers that match your
       document.
    2. Using GenCode™ saves time:
         During document preparation
         During the ''graphic output'' phase, i.e. typesetting.
    3. Preparing a document with GenCode™ will make its production more
       economical.
   This is just a short list, but it's more than enough to convince you to
start using GenCode™ in preparing your documents.
```

EXAMPLE 1.
Text File
With No Coding

```
                      ◆ Reasons for Using GenCode™ ◀

▶ There are a number of reasons for using GenCode™. They are as follows: ◀
 ▶ 1. ➡ It is easy because you can use Generic Identifiers that match your
        document. ◀
 ▶ 2. ➡ Using GenCode™ saves time:
        ➡ ➡ During document preparation ◀
        ➡ ➡ During the ''graphic output'' phase, i.e. typesetting. ◀
 ▶ 3. ➡ Preparing a document with GenCode™ will make its production more
        economical. ◀
▶ This is just a short list, but it's more than enough to convince you to
start using GenCode™ in preparing your documents. ◀

Legend
◆ Center
◀ Carriage Return
▶ Tab
➡ Indent
```

EXAMPLE 2.
Text File
With Word-Processor Coding

In case of conflict between any statement in any article and the standard, the standard governs.

Pattern	GenCode™	Typographic Parameters
1. A line centered and bold with both cancelled at the end	**⟨h1⟩/⟨/h1⟩**	**TP-1**
2. Tab code.	**⟨p⟩**	**TP-2**
3. Underline becomes emphasis code.	**⟨e1⟩/⟨/e1⟩**	**TP-3**
4. Quotes, pattern recognition used to interpret word processing quotes.	**⟨q⟩/⟨/q⟩**	**TP-4**
5. Space followed by any number by period by word processing indent code.	**⟨list⟩**	**TP-5**
6. Same pattern as #5 after first **⟨list⟩** is found.	**⟨li⟩**	**TP-6**
7. Word processing indent code (paragraph for list items).	**⟨p⟩**	**TP-7**
8. No word processing indent code (within a **⟨list⟩**).	**⟨/list⟩**	**TP-8**

EXAMPLE 3.
Pattern Recognition

Coding

```
⟨story⟩
⟨h1⟩        Reas. . . .                                 ⟨/h1⟩
⟨p⟩         There are. . . .
⟨list⟩
⟨li⟩        It is. . . .
⟨li⟩        Using. . . .
⟨p⟩         During. . . .
⟨p⟩         During. . . .       ⟨q⟩ graphic output      ⟨/q⟩
⟨li⟩        Preparing. . . .
⟨/list⟩
⟨p⟩         This is just. . . . ⟨e1⟩ your               ⟨/e1⟩
⟨/story⟩
```

EXAMPLE 4.
Text File
With GenCode™ Coding

In case of conflict between any statement in any article and the standard, the standard governs.

Abbreviation		**Typographic Parameters**
TP-1	Heading Level 1	Quad Center Memphis/Bold Point Size 12/14 Measure 34 picas Run-over lines, Quad Center
TP-2	Paragraph Text	Indent 2 em spaces to start Set following text Memphis/Medium Point size 10/12 Measure 34 picas Run-over lines, Zero indent
TP-3	Level 1 Emphasis	Italic face at point size of running text in which it occurs
TP-4	Quotes	Apply opening quotes after first delimiter. Apply closing quotes before second delimiter
TP-5	List Item, First	Indent 4 em spaces If item number <10, set 1 en space ‘‘If item number ⩾ 10, go to next step Set item number and period (=em) 1 en space Set following type Memphis/Medium Point Size 10/12 Measure 20 picas Run-over lines, Zero indent from first line
TP-6	List Item, Subsequent	Must be identical to TP-5
TP-7	List Item, Subparagraph	Indent 8 em spaces Set following type Memphis/Medium Point Size 10/12 Measure 20 picas Run-over lines, Zero indent from first line

EXAMPLE 5.
Typographic Specifications

In case of conflict between any statement in any article and the standard, the standard governs.

Reasons for Using GenCode™

There are a number of reasons for using GenCode™. They are as follows:

1. It is easy because you can use Generic Identifiers that match your document.
2. Using GenCode™ saves time:
   During document preparation
   During the "graphic output" phase, i.e. typesetting.
3. Preparing a document with GenCode™ will make its production more economical.

This is just a short list, but it's more than enough to convince you to start using GenCode™ in preparing *your* documents.

EXAMPLE 6.
Text File
Set to Typographic Specifications

```
[hd1]
   ⟨
[txt]⟨
[p]⟨
   ⟨
[nl]⟨
1.
                     ⟨
2.
                     ⟨
3.
                     [n1]
   ⟨
[txt]
[p]
```

EXAMPLE 7.
Text File
With Typical Publisher's Coding

In case of conflict between any statement in any article and the standard, the standard governs.

Document Processing at Boeing Computer Services

Sharon Adler

Manager, Production Services
Boeing Computer Services

Member		
ANSI		X3J6
Vice-Chair		
ANSI	Task Group	X3J6.1
EG CLPT	Task Group	SGML

Material prepared for this publication
1983 June 15

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the GCA copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Graphic Communications Association. To copy otherwise, or to republish, requires a fee and specific permission. © 1983 GCA

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

In the Age of Information Processing, companies and government agencies have found themselves facing critical needs for the rapid movement of information, instant update, presentation in various formats and on various devices. Managers are constantly challenged by the proliferation of systems and devices, and the demands for increased productivity and versatility. In this environment, how can one cope with change and still maintain control? These and other requirements have not been met easily by traditional systems dealing with the publication of documents.

Background

In 1978, Boeing Computer Services' software documentation group began to grapple with these issues. Boeing Computer Services, a remote computer services company, develops and publishes about 500–1,000 pages per month. These pages must be constantly reviewed and modified in draft-copy form during the development cycle. Prior to 1978, Boeing manuals were typeset using traditional methods at a cost ranging from $40/page to $100/page, with production time ranging from three to six months. This was mainly due to the large number of changes which had to be accommodated during software and documentation construction.

At Boeing Computer Services, we dealt with the problem initially by using composition software which handled only fixed character and line spaces to drive a high-speed line printer. The ''solution'' in itself caused many new problems. The numbers of pages in our books increased by 35%, increasing printing, handling, and mailing costs. In addition, the quality of the documents no longer met Boeing's standards.

Objectives

Management determined that a better approach was required. A plan was developed outlining a preliminary set of objectives.

The objectives of a new publishing system were centered on three main areas: (1) Improved productivity; (2) Decreased costs; (3) Improved quality of final copy.

The interchange of information among remote locations, using a variety of input and output devices and processing systems established critical requirements.

Word processing interfaces or conversion hardware were only partial, if not imperfect, solutions. Formats had to be modified easily; a single editorial and production system had to be developed so that both professionals and production personnel had a single ''easy-to-use'' system. The complex production process had to be streamlined. Subsequent insertion of typesetting format information was not acceptable either as a short-term or a long-term solution.

In addition, constraints existed which placed significant burdens in choosing a system. It had to run on an IBM mainframe; it had to support multiple input/output devices; and procurement could not involve file management software which was thought to be redundant in the remote computer services environment.

Several systems were evaluated. Through our attendance at GenCode™ Conferences from the Graphic Communications Association, we learned of the merits of the generalized markup concept and determined that this would form the nucleus of our system.

Generalized Markup

Before computerized composition systems were developed, typographers or editors had to ''mark up'' manuscripts. This markup was information or instructions added to the text so that it could be processed. Most often these were specific formatting instructions. Specific markup includes depressing a center or tab key on a word processor, or font, point size, and leading information on a typesetter. With this type of markup, the intellectual content of the document is often obscured. For example, both a headline and figure caption may be centered, all caps and underscored. How do you differentiate between the two for some process other than typesetting? You don't.

With generalized markup, each element of a document is tagged with an identifying code which allows it to be processed for whatever purpose is required. These tags are not format, device, or system dependent. Document elements—headlines, paragraphs, words—are tagged according to what they are, not how they look. Essentially, generalized markup allows the user to separate the text or data from the application.

How does a company develop a system based on generalized markup? The following is a brief description of some of the steps taken by Boeing Computer Services in developing an automated publishing system.

- **Standards Development**
  Generalized markup is formally defined for each type or class of document according to specific sets of standards. These standards are a consistent, organized way of presenting information. They include identifying the necessary components, the order or sequence in which used, and the form in which they are to be presented.
- **Document Analysis**
  Document analysis is the second step. During document analysis, the analyst breaks down each type of document into its component elements and establishes the relationship between these elements. Once the elements are identified and the relationships are established, the elements are linked to formatting specifications on standards.
- **Text Processing**
  In this step the process becomes system-dependent. A programmer must write a program or procedure in a formatting language which processes the markup and text. The actual text files are independent of the processing system since all files created in this manner can be pro-

cessed by many formatting languages. The text programmer must work closely with the document analysts and designers and be able to express the structure and hierarchy of the elements to the processing system.

The input for a generalized-markup-based system, therefore, can be created on any input device, with the generalized tags keyed as text. In addition, one text file marked up with these tags can produce output on laser printers, photocomposers, display terminals, or used as input to a data base management system. The benefits of such a system are many—dramatic productivity gains, costs savings, total flexibility, and a single system usable by all professionals, writers, engineers, data processors, text entry specialists. Never again will we have to rekey our text to use it for a different application.

In case of conflict between any statement in any article and the standard, the standard governs.

Comparison of Generic, Mnemonic, and Native-Mode Coding

Joseph V. Gangemi

Vice President, Systems and Programming
Lehigh/Rocappi, Inc.

Member

ANSI		X3J6
ANSI	Task Group	X3J6.1

Material originally presented at the
Research & Engineering Council Conference
Chicago, 1983 May 3

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the GCA copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Graphic Communications Association. To copy otherwise, or to republish, requires a fee and specific permission. © 1983 GCA

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

Generic Coding is a term being used in several different contexts and has been applied to many different coding schemes. However, I would like to address the topic of generic coding as a *concept*, not a set of codes. By definition, generic coding is a methodology by which the intellectual content of data is incorporated into a file in machine processable form. It is not a specific set of codes or a processing or programming language.

The intent of a generic coding system is not to control format but to capture the identification of the constituent data elements along with the data itself. The byproduct of the system is the ability to use the data in a broad range of applications. Applications may be format-oriented where data is being presented on such visual media as paper or a VDT display tube, or content-oriented where data is being interrogated and analyzed on line.

Many systems confuse generic coding with other coding methodologies. Two alternate coding schemes should be considered to contrast some basic differences in the philosophies.

The first alternative scheme (and most closely aligned with generic coding) is ***mnemonic coding***. This scheme is sometimes referred to as a generic coding scheme, but there are significant differences between the two concepts.

Mnemonic codes are a set of tags inserted in a data file at points at which an event must occur. An event is any change in format or effect which must be identified to an application program for processing purposes. They have no predetermined function. Their purpose is to be a placeholder in the data stream at which point some control can be applied.

Mnemonic codes are defined at run time to control the format of the output product. They can have *some generic scope, but it is a* ***secondary*** *consideration,* ***not a primary*** function.

Another alternative coding scheme is ***native mode coding***. Native mode codes are those which are application and system dependent. They are established for a single purpose and perform a specific predetermined function. While even these codes can be generic within the scope of the application, it is *usually* ***very difficult, if not impossible,*** *to use them in other application or system environments.*

What should a ***true generic coding system*** deliver?

1. *Separation of data from any individual application.* There should be complete portability among systems so that data can be moved freely from one environment to another with no constraints.
2. *Unlimited and diversified use of data in both format dependent* applications such as computer printouts (including laser output and COM), VDT's, and computer typesetting, *and content dependent applications* such as information retrieval, query and analysis, electronic publishing (including VIDEOTEX).

How do you *implement a generic coding system?*

1. *Substantial Preplanning*—Like any computer-oriented system, you must think it through and plan every aspect. You must analyze your data requirements and establish standards and rules. At present, this planning process is only cost effective on large projects which can support this overhead, or on a range of information which is consistent in content to utilize the same system, or on repetitive products which recover the cost over several iterations. Small, one time applications or data with broad content or format variations may not be cost effective.
2. *Enforced Discipline*—The standards and rules established for the system give you extremely effective control of the information captured in the database. This discipline produces consistent and predictable data in the database. The result should be higher quality end products from the information base.

At this time, there are no true industry standards to use as a foundation for establishing a generic coding system on a broad base. While several users have developed localized systems, either internal to an organization or within a small community of compatible users, there are still limitations on the widespread use of the data.

An industry standard would help to establish a method of communicating the specifications of a database in clear and unambiguous terms. It would also help direct the development efforts of both hardware and software suppliers so that generically coded files result from and are processable by their systems and applications.

Two standards efforts currently in progress, and in which I am a participant, are the Text Description Language Task Group (concurrently ISO TC97/SC5 EGCLPT/SGML and ANSI X3J6.1) and the GCA GenCode™ committee. The ISO/ANSI committee is developing a standard mark-up language for use in a text database. The GenCode™ committee is developing document type descriptions for use in book publishing based on the language syntax of the proposed international standard.

Does a generic coding system reduce costs of data capture and usage? The answer is not specific since many factors affect cost. In general, I would have to say it will. For example, if data is captured once at the source and can be passed on fully coded to a supplier, the supplier's costs must be lower since he does not have to input the data. If a single system is in effect for a broad range of applications, training costs must be lower since only one system of data capture must be learned.

Cost savings come from different areas depending on the type of application, the support available from suppliers providing the services, and from the type of product being produced.

In the final analysis, ***generic coding is the method by which we can truly make the text content a product unto itself***. As we move more into the computer communications era, we should find new opportunities to use this product to fulfill both our revenue and utilitarian needs.

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

GCA

CONTRIBUTORS

GCA

Graphic Communications Association
1730 North Lynn Street / Suite 604 • Arlington, Virginia 22209 • Phone (703) 841-8160

An Affiliate of Printing Industries of America, Inc. PIA INC

[BLANK PAGE]

Contributors

Table of Contents

1. **ANSI/ISO Committee Participation**

a. Officers 255
b. Members 256
c. Task Group Members 259

2. **Authors' Biographies**

a. Sharon Adler 260
b. Aaron H. Bigman 260
c. William W. Davis, Jr. 260
d. Joseph V. Gangemi 260
e. Jordan J. Glogau 260
f. Charles F. Goldfarb 260
g. Joseph P. Johnson 261
h. James D. Mason 261
i. Brian L. Murray 261
j. Donald H. Rollert 261
k. John A. Šullivan 261
l. C. ''Kit'' von Sück 261

In case of conflict between any statement in any article and the standard, the standard governs.

[BLANK PAGE]

PART C

CONTRIBUTORS

1. ANSI/ISO Committee Participation

The development of the standard to the point of submission of the

Sixth Working Draft, International Standard
ISO TC97/SC5/EG CLPT-X3J6 N177
1983 Jun 01

to the International Organization for Standardization (ISO) has been made possible by the personal contributions of energy, talent, and time of the individuals listed below, and by the support of their parent organizations.

The standard development process took place under two coexisting Committees:

American National Standards Committee
operating under the procedures of the
American National Standards Institute

X3—Information Processing Systems
J6 —Computer Languages for the Processing of Text

Project 203
X3J6.1 Task Group
Text Description Language

International Organization for Standardization (ISO)

TC97 —Information Processing Systems
SC5 —Programming Language
EG —Experts Group
CLPT—Computer Languages for the Processing of Text

Project 97.5.12
EG CLPT Task Group
Text Description Language

a. Officers

The officers and functional representatives who have led this effort include:

Chairperson (non-voting):

Charles D. Card
Sperry

Vice Chairperson:

Grant Crawford
Computing Services
University of Alberta

Secretary:

Ann Marie Curley
Wayne State University

International Representative (former):

Eric H. Clamons
HIS Incorporated

b. Members

The Officers listed above and the Members, present and past, listed below have contributed to the development of the whole standard.

Sharon Adler
Boeing Computer Services

Jose M. Baeza
US Army Communications Command

Lawrence A. Beck
Grumman Data Systems

Robert W. Bemer
Bemer Software, Inc.

Aaron H. Bigman
Standard & Poor's Corporation

James Cox
IBM Corporation

Earl Culham
Computing Services
University of Alberta

William W. Davis, Jr.
Internal Revenue Service

Blanton C. Duncan
Consultant

Joseph V. Gangemi
Lehigh/Rocappi, Inc.

Terry Girill
National Magnetic Fusion Energy
Computer Center
Lawrence Livermore National Laboratory

Charles F. Goldfarb
IBM Corporation

Randall S. Groves
Boeing Computer Services

Blenda Hildebrandt
Sperry

Charles P. Howerton
National Bureau of Standards
Boulder Laboratories

Walter Lapinski
Burroughs Corporation

Charles Lightfoot
Autologic, Inc.

Clement Luk
National Magnetic Fusion Energy
Computer Center
Lawrence Livermore National Laboratory

J. Sperling Martin
Aspen Systems Corporation

James D. Mason
Oak Ridge National Laboratory

Norman Meyrowitz
Brown University

Elizabeth K. Michael
TYMSHARE

Alan Mink
National Bureau of Standards

John C. Oberholzer
Sperry

George Oerter
Sperry

Louis O'Korn
Chemical Abstracts Service

Robert Pariseau
PRIME Computer

Lynne A. Price
CALMA Research & Development

Robert E. Rountree
National Bureau of Standards

Norman W. Scharpf
Graphic Communications Association

Diane L. Smolenski
Sperry

Edwin J. Smura
Xerox Corporation

Sandy Sparks
Lawrence Livermore National Laboratory

David W. Thompson
Lawrence Livermore National Laboratory

Robert Travis
Digital Equipment Corporation

William W. Tunnicliffe
Graphic Communications Association

E. J. Wallman
HIS Incorporated

Shirley Ward Watkins
National Bureau of Standards

William A. Whitaker
McLean, VA

Roger Willis
Digital Equipment Corporation

Allan R. Wylie
U.S. Air Force

[BLANK PAGE]

c. Task Group Members

Members of the Text Description Language Task Group, present and past, who have made contributions particularly intended for Part Six: Document Markup Metalanguage include:

1. Officers

Chairperson

William W. Davis, Jr.
Internal Revenue Service

Vice Chairperson

Sharon Adler
Boeing Computer Services

Secretary

Lawrence A. Beck
Grumman Data Systems

Editor

Charles F. Goldfarb
IBM Corporation

2. Members & Consultants

Aaron H. Bigman
Standard & Poor's Corporation

Robert W. Bemer
Bemer Software, Inc.

James Cox
IBM Corporation

Joseph V. Gangemi
Lehigh/Rocappi, Inc.

Michael Goodfellow
IBM Corporation

Randall S. Groves
Boeing Computer Services

Walter Lapinski
Burroughs Corporation

Charles Lightfoot
Autologic, Inc.

J. Sperling Martin
Aspen Systems Corporation

Bettie McDavid Mason
Consulting Editor

James D. Mason
Oak Ridge National Laboratory

George Oerter
Sperry

Louis O'Korn
Chemical Abstracts Service

Norman W. Scharpf
Graphic Communications Association

Diane L. Smolenski
Sperry

Edwin J. Smura
Xerox Corporation

William W. Tunnicliffe
Graphic Communications Association

C. ''Kit'' von Sück
Digital Equipment Corporation

Authors' Biographies

a. **SHARON ADLER** is Manager of Production Services in the Publications Planning & Development Department of Boeing Computer Services. Her duties include development of a text processing and photocomposition system based on the generalized markup concept. Her entire career has been in printing and publishing with focus the last 8 years on computerized composition systems. As production manager and systems designer at Duke University, she designed and implemented a data base system for text management and photocomposition of university course catalogs and telephone directory. Ms. Adler is Vice Chair of ANSI X3J6.1 Committee on Information Processing Systems, Programming Languages, Text Interchange and Processing; Chairman of GCA's GenCode™ Committee; and Group Manager of End User Interface Group at SHARE, for users of large IBM computing systems. Ms. Adler earned her BS from the University of Wisconsin and did graduate work in computer sciences at Duke University.

b. **AARON H. BIGMAN** is a Senior Financial Publishing Consultant for Standard & Poor's Corporation in New York City. He was formerly the Vice President of Information Technologies for GCA. Bigman's experience lies in the area of data base management and publishing systems, and he has worked for both government and industry on large-scale publishing systems and applications. Mr. Bigman holds a B.S. in Business Administration from the University of Maryland and has done graduate work at the University of Maryland and American University. Previous to joining GCA, Mr. Bigman worked for Sterling Systems, Inc., where he worked on the development of print-on-demand systems for the publications branch of the U.S. Navy. Prior to that, he was a management analyst for the Federal Railroad Administration, U.S. Department of Transportation.

c. **WILLIAM W. DAVIS, JR.** is the coordinator for all Internal Revenue Service (IRS) publishing applications in the fields of electronic typesetting, computerized text processing, text-oriented data base management, and on-demand printing and electronic forms composition. He is also leading efforts to develop a fully automated management information system for publishing management in the IRS. Mr. Davis graduated from the Rochester Institute of Technology in 1965 with a B.S. in Printing Management. He began working at the IRS in 1965 as a printing management intern and in 1969 became responsible for the development of IRS's electronic composition program. Mr. Davis is chairman of X3J6.1, which is concerned with Text Document Metalanguage, and is a member of both the national and international committee developing the Computer Languages for Text Processing. He has been working with the GCA GenCode™ Committee in the development of GenCode™ since its inception and has been a speaker at a number of GCA seminars and conferences involving tabular composition, loose-leaf publishing, and GenCode™.

d. **JOSEPH V. GANGEMI** Vice President of Systems and Programming, Lehigh/ROCAPPI, Inc. studied electrical engineering at Drexel Institute in Philadelphia. Since 1967 he has been the principle architect of all of ROCAPPI'S software, including the ROCAPPI Typesetting System, the ROCAPPI Data Base Management Service, and the ROCAPPI Network System for Telecommunications. He is responsible for all research and development of ROCAPPI programs and systems and for supervision of all programming personnel. Mr. Gangemi is a member of the GCA GenCode™ Committee and a member of the Text Structures and Interchange Task Group of ANSI X3J6, which is developing standards for a markup language for the description and interchange of text data bases. He is also the author of several articles dealing with generic coding and text markup concepts.

e. **JORDAN J. GLOGAU** Executive Vice President of Medianet, Inc., has been involved with data processing aspects of the graphic arts industry for ten years. He has designed and coordinated complex data base and typesetting projects, including work in systems analysis, programming, marketing, financial, purchasing, applications, and sales. Mr. Glogau received a B.S. in Printing Technology from the Rochester Institute of Technology and has taken graduate courses in accounting and data processing. Before joining Medianet, Inc., Mr. Glogau worked for the New England Typographic Service, Charles P. Young, Bedford Computer, Digital Equipment, and the Internal Revenue Service.

f. **CHARLES F. GOLDFARB** is a Senior Analyst in IBM's San Jose Research Laboratory, with responsibilities in the areas of strategy and technical direction for text processing and office systems. He has contributed to a number of IBM text processing and word processing products—most recently, the Document Composition Facility—and was the leader of the research project that invented the Generalized Markup Language (GML) in 1969. He has also served as a market planner, developing market research and business justification methodologies for publishing systems. Dr. Goldfarb is a member of the text processing standards committees of ANSI (X3J6) and ISO (TC97/SC5/EG CLPT). He is a past chairman of the working group that is developing the Standard Generalized Markup Language, and has been the editor of the draft standard since the inception of the work. He is a graduate of Harvard Law School and Columbia College and a member of the California Bar.

g. **JOSEPH P. JOHNSON** is Technology Assessment Officer for the Federal Deposit Insurance Corporation (FDIC). His current technical duties include technology assessment and advice in the areas of distributed processing, local area networks, text processing, graphics, microcomputers, and data base software. Mr. Johnson previously worked with the Department of Health, Education, and Welfare (DHEW), now the Department of Health and Human Services (DHHS). His technical duties included the analysis, design, and programming of computer systems in the areas of biomedical and mathematical statistics, data processing, and information storage and retrieval. Mr. Johnson has taught a computer science in the graduate school at the United States Department of Agriculture for 15 years. He is a member of the Association of Computing Machinery, IEEE Computer Society, and the GenCode™ Advisory Committee to the ANSI X3J6 Committee.

h. **JAMES D. MASON** is a text-management development and planning specialist in the Technical Publications Department at Oak Ridge National Laboratory and serves as the Laboratory's word-processing coordinator. He received a B.A. in Mathematics and Physics and a M.A. in English from Vanderbilt University and a Ph.D. in English and German from the University of Tennessee. Dr. Mason is Secretary of ANSI X3VI, Office Systems, and a member of ANSI X3J6, Computer Languages for Processing of Text.

i. **BRIAN L. MURRAY** is Director of the Text Editing and Photocomposition Branch of the Canadian Government Printing Office. This section is responsible for the automation of text processing and typesetting for all publications generated by the Canadian legislature. The current system accepts data on-line from terminals located in all legislative branches, and supports both optical character recognition (OCR) equipment and a variety of word processing devices. Mr. Murray has worked on the development of automated pagination programs; enhancement of hyphenation-and-justification programs; preprocessing test inputting routines; and systems necessary to evolve from a paper-tape input system to the present ATS online and edit system.

j. **DONALD H. ROLLERT** has been Manager of the Graphic Systems Development Division of the Government Printing Office (GPO) for the past eight years. This division is responsible for the design and development of the automated composition and full text data base management systems at GPO. Before joining GPO he was employed with Mergenthaler Linotype Company for 13 years, the last six of which he held the position of Manager in the graphic systems engineering department, where he was involved with the Linotron 1010 contract with GPO. Mr. Rollert is a graduate of the U.S. Coast Guard Academy, where he received a B.S. in Engineering.

k. **JOHN A. SULLIVAN** is Head of the Composition Software Section of the Canadian Government Printing Office. This section, which is currently investigating generic coding to accommodate efficiently the many devices used to input text, is responsible for the development and implementation of composition software used to produce publications.

l. **C. "KIT" VON SÜCK** has been involved with the graphic arts industry for the past 16 years. He is currently Program Manager of Digital Equipment Corporation's Document Systems and Applications Group (DSAG). For the past three years he has worked at Digital developing the educational and training materials for DECset. Previous to joining Digital, Mr. von Sück owned and operated a commercial printing business and taught offset lithography, letterpress, and screen printing. He attended Keene State College, where he majored in Industrial Education with a primary focus on the printing trades.

NOTES

NOTES

NOTES

ORDER FORM for the GCA STANDARD 101-1983

Please send _______ copies @ $ _______ = $ _______ (Total Cost)

Single Copy Price **$48.00**

Quantity Discounts

10-49 copies	less 10%
50-99 copies	less 20%
100 or more copies	less 33-1/3%

GCA members may ***deduct 25%*** from the net price. (Membership in PIA does not include membership in GCA.)

Printing companies which are not members of PIA must ***add 10%*** to the net price.

All orders must be prepaid. Make check payable to Graphic Communications Association.

Where possible please provide a street address so your order may be shipped via United Parcel Service if necessary.

Foreign orders will be shipped surface rate. Add $9.96 per book for postage and handling for shipment via "Air Other Rate."

Name

Title

Company

Address

City

State Zip

Telephone

Graphic Communications Association
An Affiliate of Printing Industries of America, Inc.

ORDER FORM for the GCA STANDARD 101-1983

Please send _______ copies @ $ _______ = $ _______ (Total Cost)

Single Copy Price **$48.00**

Quantity Discounts

10-49 copies	less 10%
50-99 copies	less 20%
100 or more copies	less 33-1/3%

GCA members may ***deduct 25%*** from the net price. (Membership in PIA does not include membership in GCA.)

Printing companies which are not members of PIA must ***add 10%*** to the net price.

All orders must be prepaid. Make check payable to Graphic Communications Association.

Where possible please provide a street address so your order may be shipped via United Parcel Service if necessary.

Foreign orders will be shipped surface rate. Add $9.96 per book for postage and handling for shipment via "Air Other Rate."

Name

Title

Company

Address

City

State Zip

Telephone

Graphic Communications Association
An Affiliate of Printing Industries of America, Inc.

ORDER FORM for the GCA STANDARD 101-1983

Please send _______ copies @ $ _______ = $ _______ (Total Cost)

Single Copy Price **$48.00**

Quantity Discounts

10-49 copies	less 10%
50-99 copies	less 20%
100 or more copies	less 33-1/3%

GCA members may ***deduct 25%*** from the net price. (Membership in PIA does not include membership in GCA.)

Printing companies which are not members of PIA must ***add 10%*** to the net price.

All orders must be prepaid. Make check payable to Graphic Communications Association.

Where possible please provide a street address so your order may be shipped via United Parcel Service if necessary.

Foreign orders will be shipped surface rate. Add $9.96 per book for postage and handling for shipment via "Air Other Rate."

Name

Title

Company

Address

City

State Zip

Telephone

Graphic Communications Association
An Affiliate of Printing Industries of America, Inc.

GCA

STANDARD 101-1983

Document Markup Metalanguage

Adopted by
DEPARTMENT OF DEFENSE
10 August 1983

GENCODE™

And The

STANDARD GENERALIZED MARKUP LANGUAGE

(SGML)

Graphic Communications Association
1730 North Lynn Street / Suite 604 • Arlington, Virginia 22209 • Phone (703) 841-8160

An Affiliate of Printing Industries of America, Inc. PIA INC